UiO : Department of Informatics
University of Oslo

Genome Assembly: Scaffolding Guided by Related Genomes

Runar Furenes
Master's Thesis, Spring 2013

Genome Assembly: Scaffolding Guided by Related Genomes

Runar Furenes

May 2, 2013

Abstract

Genomic research relies on computers to process large amounts of genomic data. In order to digitize such data, the genomes have to be sequenced and assembled. Modern sequencing technologies allow fast and inexpensive sequencing.

Sequencing machines produce multiple chunks of sequences called *reads*, which are assembled into *contigs*, and then further into larger pieces called *scaffolds*. The process of scaffolding contigs often requires obtaining additional data through lab work, which is both time-consuming and expensive.

The purpose of this thesis is to assess whether contigs can be scaffolded with the aid of previously sequenced related genomes, and whether the use of *multiple* related genomes can increase the precision of the resulting scaffolds.

A pipeline with a simple, prototypical algorithm was developed to process contigs using information from related genomes. This pipeline produces scaffolds and provides an evaluation of these.

Contigs from 4 bacterial sequencing projects were scaffolded with 10 related genomes as guides for each bacterium.

The results showed that using multiple guiding genomes, which were closely related to the target genome, enabled scaffolds to be produced with few errors.

Acknowledgements

First and foremost I would like to thank my supervisor *Torbjørn Rognes* for his time, help and patience through the process of writing this thesis.

Secondly I would like to thank *Alexey A. Gritsenko* for handing me the datafiles I needed, and *Lex Nederbragt* for introducing me to MUMmer.

I would also like to thank *Nils Damm Christophersen* for helping me see the bigger picture and *Geir Kjetil Sandve* for holding motivating weekly meetings.

I am very grateful for *Emanuele Lapponi* telling me what *not* to do and *Tobias G. Waaler* for code reviewing and always being helpful.

I am also very grateful to all my fellow students at the 10th floor in *Ole Johan Dahls hus* for the table tennis, the technical discussions and the laughs.

I would also like to thank *Dag Langmyhr* for his quick responds to all my LaTeX issues, and *Sean Erik Scully* for language assistance.

A very special thanks to *Marit Enny Gismarvik* for proofreading and for always believing in me.

Runar Furenes
University of Oslo
May, 2013

Preface

This thesis is written with a master's student of bioinformatics as the imagined reader. Basic knowledge of informatics is therefore assumed, and only rudimentary prior knowledge of biology is necessary.

My project started out with different experiments with genome assembly and scaffolding. A semi-complete program was made using mate-pairs to build scaffolds, including a stochastic optimization algorithm.

This approach turned out not to be any better than the built-in scaffolding modules in various assembly software, which used more sophisticated methods.

A new approach to scaffolding was therefore taken, using related genomes as guides, without the use of mate-pair information. If this could produce scaffolds with low error rates, it could be used to save time and funding needed for the extra lab-work.

The resulting pipeline of guided scaffolding is to be considered a *prototype*, where several simplifications are taken. It works as a proof of concept of what is possible when multiple related genomes are involved in a scaffolding process, also when these genomes are from different species.

Contents

1 Introduction **1**
- 1.1 Motivation for the thesis . . . 1
  - 1.1.1 Online genome availability . . . 1
  - 1.1.2 Prototype for using multiple genomes in scaffolding . . . 1
- 1.2 Hypotheses . . . 2

2 Background **3**
- 2.1 Biology . . . 3
  - 2.1.1 Genome . . . 3
  - 2.1.2 DNA and RNA . . . 3
  - 2.1.3 RNA codons and proteins . . . 4
  - 2.1.4 Genes . . . 5
  - 2.1.5 Translation and reading frames . . . 6
- 2.2 Bioinformatics . . . 6
- 2.3 Genome assembly . . . 7
  - 2.3.1 Sequencing . . . 9
  - 2.3.2 Contig assembly . . . 11
  - 2.3.3 Scaffolding . . . 12
- 2.4 Sequence alignment . . . 13
  - 2.4.1 Nucleotide level . . . 13
  - 2.4.2 Protein level . . . 14
- 2.5 Data formats . . . 14
  - 2.5.1 FASTA . . . 14
  - 2.5.2 FASTQ . . . 15

3 Methods **17**
- 3.1 Overview . . . 17
  - 3.1.1 Hypothesis on optimal contig end length . . . 17
  - 3.1.2 Nucleotide- and protein alignment in guiding genomes 20
- 3.2 Description of *GuideScaff* . . . 20
  - 3.2.1 Choosing guiding genomes . . . 20
  - 3.2.2 Contig preparation and contig end extraction . . . 21
  - 3.2.3 Aligning contig ends to guiding genomes . . . 21
  - 3.2.4 Contig links creation . . . 22
  - 3.2.5 Creating final scaffold file from contig links . . . 25
- 3.3 Implementation . . . 26
  - 3.3.1 Additional speed gain with GNU Parallel . . . 28

4 Evaluation **29**
4.1 Commonly used evaluation metrics 29
4.1.1 N50 29
4.1.2 Breakpoints 29
4.1.3 Modifications to breakpoint counting 31
4.1.4 Measure of correctness 31

5 Materials **33**
5.1 Target genomes and contigs 33
5.2 Guiding genomes 36

6 Results **41**
6.1 Initial runs of *GuideScaff* on all datasets 41
6.1.1 Using entire contigs 41
6.1.2 Choosing an optimal contig end length 42
6.1.3 Using contig ends with a fixed length 43
6.2 Using multiple guiding genomes 48
6.2.1 Without contig ends extraction 48
6.2.2 With contig ends extraction 53

7 Discussion **57**
7.1 Interpreting the results 57
7.1.1 Initial runs on all datasets 57
7.1.2 Extracting contig ends 57
7.1.3 Using multiple guiding genomes 58
7.2 Analysis of the proposed method 58
7.2.1 Performance 58
7.2.2 Potential usage 58
7.2.3 Weaknesses 59
7.3 Further work 59
7.4 Conclusion 60

A Mauve genome similarity plots **67**

B Result tables **73**

Chapter 1

Introduction

1.1 Motivation for the thesis

The process of *genome assembly* is necessary prior to any computational analysis of genomic data, by gathering and digitally representing biological DNA.

To create a digital representation of the DNA sequences, the following steps are followed:

- *reads* are produced with a sequencing machine
- *contigs* are created from the reads with an assembler
- *scaffolds* are created from the contigs by an assembler or a different program

In scaffolding, the ordering, orientation and distances of contigs are determined. There exists several methods of scaffolding, and most rely on mate-pairs, which requires additional lab-work. This is both time consuming and expensive.

A *reference genome* is an already sequenced genome of the same species as the organism to be sequenced. If a *reference genome* is available, this may help in the process of scaffolding.

The idea of this thesis is to explore the feasibility of using this information for scaffolding, and to assess whether this could replace expensive and time consuming lab-work.

1.1.1 Online genome availability

Online databases of fully sequenced genomes are continuously growing. The National Center for Biotechnology Information (NCBI) [31] has for instance complete records of 188 eukaryotes, 2,677 prokaryotes and 3,511 viruses at the time this thesis is written.

1.1.2 Prototype for using multiple genomes in scaffolding

A prototype method named *GuideScaff* was implemented and tested with 4 different datasets. *GuideScaff* processes a set of contigs and a set of related

genomes. It then produces a set of scaffolded contigs. It also gives an evaluation to assess the quality of the resulting scaffolds.

To avoid confusing with the common meaning of a *reference genome*, the term *guiding genome* is used in this thesis, meaning an already sequenced genome *related* to the genome at hand.

1.2 Hypotheses

The following hypotheses are explored within this thesis:

1. Guiding genomes can be helpful in scaffolding
2. Many guiding genomes are preferable to a few
3. It can be beneficial to align only the *ends* of contigs

When genome assemblers produces an unsatisfactory amount of contigs or scaffolds, is it possible to further enhance the scaffolds with the aid of guiding genomes?

Assuming that is *is* beneficial to use guiding genomes in scaffolding, can many guiding genomes be preferable to only one or a few?

Also assuming that the contigs a genome assembler produces are 100% correct, the scaffolding process should only be concerned with linking contigs. If contigs contains sequences with multiple matches in the guiding genomes, this ambiguity may cause errors in the scaffolds. However, if only the end of long contigs are mapped to guiding genomes, and they map with less ambiguity than whole contigs, could this enhance the resulting scaffolds? And if this is the case, what is a good way to extract these contig ends?

If the target genome and a guiding genome are distantly related, an aligner may have difficulties finding matches for contigs or contig ends in the nucleotide sequence of the guiding genome, as they differ too much on a nucleotide level. Proteins are more conserved than genes between related genomes [5]. If the contigs are aligned on a protein level instead, can this enable more of the contigs to be aligned to the guiding genome?

Chapter 2

Background

2.1 Biology

This section will provide the basics of molecular biology needed to understand the motivation for this thesis and the methods discussed within it.

2.1.1 Genome

A *genome* is "the complete complement of an organism's genetic material." [5] Genomes consist of one or several chromosomes, residing inside the cells of the organism. Each chromosome is a double helix of DNA molecules consisting of a string of nucleotides. A chromosome can be linear or circular, where a linear chromosome has ends, while a circular chromosome has its ends connected, forming a ring.

Cells are commonly divided into *prokaryotes* and *eukaryotes*. The methods in this thesis only considers prokaryotes, and more specifically *bacteria*.

Plasmids are sometimes present in bacteria. Plasmids are small circular units of DNA, and a part of the genome. In this thesis, chromosomes are in certain contexts used to denote both chromosomes and plasmids as they both consider units of DNA inside bacteria.

2.1.2 DNA and RNA

DNA-strands consist of a backbone of sugars and phosphate groups with *nucleotides* connected to them. There are 4 different nucleotides in DNA: guanine (G), adenine (A), thymine (T) and cytosine (C) [5]. Figure 2.1 shows how the backbones of each strand are constant, while the nucleotides varies between the 4 types. The presence and order of these nucleotides defines a genomic code, which is unique for each organism [5].

Each of the 4 nucleotides has a *complementary* nucleotide present in the other strand. Guanine (G) binds to Cytosine (C), and Adenine (A) binds to Thymine (T). Because of this complementarity, a nucleotide sequence from one strand implicitly gives the sequence of the other strand, as it will consist of the complementary nucleotide in each position.

Each of the strands in the DNA-helix has a 5'-end and a 3'-end. Reading a nucleotide sequence from the 5'-end to the 3'-end is called reading it *downstream*, while reading the sequence from the 3'-end to the 5'-end is called reading it *upstream* [5].

The two strands in the DNA helix are oriented in opposite directions. This means that if a sequence is read downstream on one strand, the sequence of the other strand upstream is the *reversed complementary* sequence.

DNA nucleotides are *transcribed* to *RNA* nucleotides. RNA consists of the same nucleotides as DNA, except Thymine (T) is replaced with Uracil (U) [5].

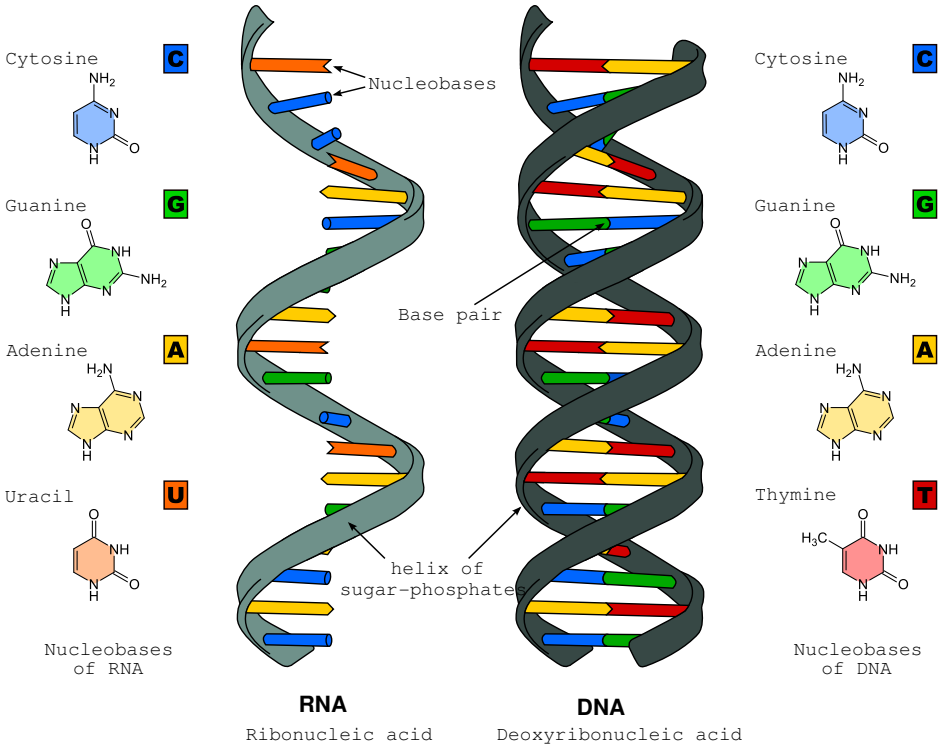

Figure 2.1: RNA and DNA molecules

Source: http://commons.wikimedia.org/wiki/File:Difference_DNA_RNA-EN.svg

The order of the nucleotides in RNA sequences acts as "recipes" for proteins. This is commonly known as "the central dogma of molecular biology" [5]. This dogma expresses how DNA is transcribed to *RNA*, which may be *translated* to chains of *amino acids*, which are *proteins*.

2.1.3 RNA codons and proteins

Proteins consist of 20 different kinds of *amino acids*. A *codon*, or *triplet*, is three consecutive nucleotides in an RNA sequence. Each codon in an RNA sequence is translated into one of the 20 amino acids. With 4 possible nucleotides, there can exist $4^3 = 64$ unique codons. Some codons code for

the same amino acid. It is therefore a many-to-one relation between codons and amino acids.

In addition, three codons acts as *stop codons*, which do not code for amino acids but stop the translation.

There is a standard genetic code which most organisms follow in the translation of RNA to proteins [5]. This is shown in table 2.1.

Amino acid	Symbol	Codons
Alanine	A	GCU, GCC, GCA, GCG
Arginine	R	CGU, CGC, CGA, CGG, AGA, AGG
Asparagine	N	AAU, AAC
Aspartic acid	D	GAU, GAC
Cysteine	C	UGU, UGC
Glutamic acid	E	GAA, GAG
Glutamine	Q	CAA, CAG
Glycine	G	GGU, GGC, GGA, GGG
Histidine	H	CAU, CAC
Isoleucine	I	AUU, AUC, AUA
Leucine	L	CUU, CUC, CUA, CUG, UUA, UUG
Lysine	K	AAA, AAG
Methionine	M	AUG
Phenylalanine	F	UUU, UUC
Proline	P	CCU, CCC, CCA, CCG
Serine	S	UCU, UCC, UCA, UCG, AGU, AGC
Threonine	T	ACU, ACC, ACA, ACG
Tryptophan	W	UGG
Tyrosine	Y	UAU, UAC
Valine	V	GUU, GUC, GUA, GUG
Stop codons	Stop	UAA, UAG, UGA

Table 2.1: Amino acids and which codons (triplets) of RNA are translated to each of them, using the IUPAC alphabet.

Source: http://www.cbs.dtu.dk/courses/27619/codon.html

2.1.4 Genes

An important concept within molecular biology is *genes*. There are several definitions of genes [13, 23]. In this thesis, a gene is considered a part of the DNA which, when transcribed to RNA, translates to *one* protein. When this happens, the gene is said to be *expressed*.

In prokaryotes, a gene starts with a start codon and ends at a stop codon. As shown in table 2.1, AUG is a start codon, while UAA, UGA and UAG are stop codons.

```
Original RNA           ACGUAUAGGUCAUACC
1st reading frame      ACG UAU AGG UCA UAC (C)
2nd reading frame  (A) CGU AUA GGU CAU ACC
3rd reading frame (AC) GUA UAG GUC AUA (CC)
                       \ / \ / \ / \ / \ /  RNA codons (triplets) are
                        |   |   |   |   |   translated to amino acids
1st amino acids         T   Y   R   S   Y
2nd amino acids         R   I   G   H   T
3rd amino acids         V   <STOP>
```

Listing 2.1: Transcription from different reading frames

2.1.5 Translation and reading frames

One RNA sequence can be translated into different proteins, depending on where the translation starts. The starting RNA nucleotide determines what is considered the first triplet. Then the next triplet will be the three consecutive nucleotides after the first triplet, and so on. This is one *reading frame*. If the initial nucleotide is shifted one position, a different reading frame is used, and all the triplets will be affected by this shift. The RNA sequence in listing 2.1 demonstrates how this can unfold. 3 different *reading frames* can be chosen, depending on where the translation starts. This can yield different amino acids, or even stop the translation completely. In the example, the 2nd codon is a stop codon when considering the 3rd reading frame.

There are 6 different reading frames, 3 for each strand of the DNA helix. Listing 2.1 shows how the 3 reading frames are considered in *one* of the strands.

Related organisms share similar sequences, as they have a common ancestor. During evolution, several changes, or *mutations*, may have occurred in each organism's genome. On a microscopic level, changes can be either one or a combination of the following:

Substitution One nucleotide may have been replaced by another

Deletion A nucleotide has been removed

Insertion A new nucleotide has been inserted

On a *macromolecular* level, entire segments (possibly genes) of DNA can be missing (deletion), added (insertion) or swapped places in the genome (rearrangement) [32].

As there are fewer amino acids than there are possible codons in RNA, few and simple modifications may or may not change the gene expression when RNA is translated to proteins. Listing 2.2 shows an example of substitutions *not* causing any effect on the protein being transcribed.

2.2 Bioinformatics

The field of bioinformatics is defined in a various of ways. Jin Xiong [32] defines it as

```
CGUAUAGGUCAUACC
CGCAUAGGUCAUACC <- U has mutated to C
CGCAUAGGUCAUACA <- C has mutated to A
 R  I  G  H  T  <- All  3 sequences translates to these amino acids
```

Listing 2.2: Mutations not causing different proteins

> "[the] discipline of storing and analyzing biological data using computational techniques. More specifically, it is the analysis of the sequence, structure, and function of the biological macromolecules — DNA, RNA and proteins — with the aid of computational tools that include computer hardware, software, and the Internet"

The scope of this thesis adheres to this definition, as it is concerned with sequences of DNA, RNA and proteins. The thesis is also concerned with comparative analysis of bacterial genomes, where computational techniques are essential.

2.3 Genome assembly

In order to work with genomic sequences on a computer the sequences needs to be gathered and represented digitally. The complete process from biological material to a digital representation of the sequence consists of *sequencing* and *genome assembly*. The process is visualized in fig. 2.2.

A useful analogy of genome assembly is a jigsaw puzzle [24]. The genome corresponds to the image on the jigsaw box and the assembly problem is equivalent to putting the pieces together in correct combinations, restoring the jigsaw image.

Biological material is prepared in the lab by replicating the DNA and shearing it into small fragments — small enough to be handled by the sequencing equipment used.

A sequencing machine processes these fragments and determines which bases are present in each fragment. An interpreted fragment from a sequencing machine is called a *read*.

The next step is to process these reads, which can happen in one of two ways: *De novo*[1] genome assembly or a *comparative* method mapping the reads to a reference genome.

De novo assembly is the process of using information from the set of reads to restore the genomic sequence partially or completely.

A comparative approach may be taken if the *target genome*[2] has been previously sequenced. If this is the case, then each read can be *mapped* to this previously sequenced genome (called the *reference genome*). Reads can be mapped to a reference by various traditional text-matching algorithms (section 2.4 on page 13), and a match can occur when there is a non-ambiguous way of placing a read on the reference. A *de novo*

[1] Latin: anew, afresh [16]

[2] The genome to be sequenced and assembled

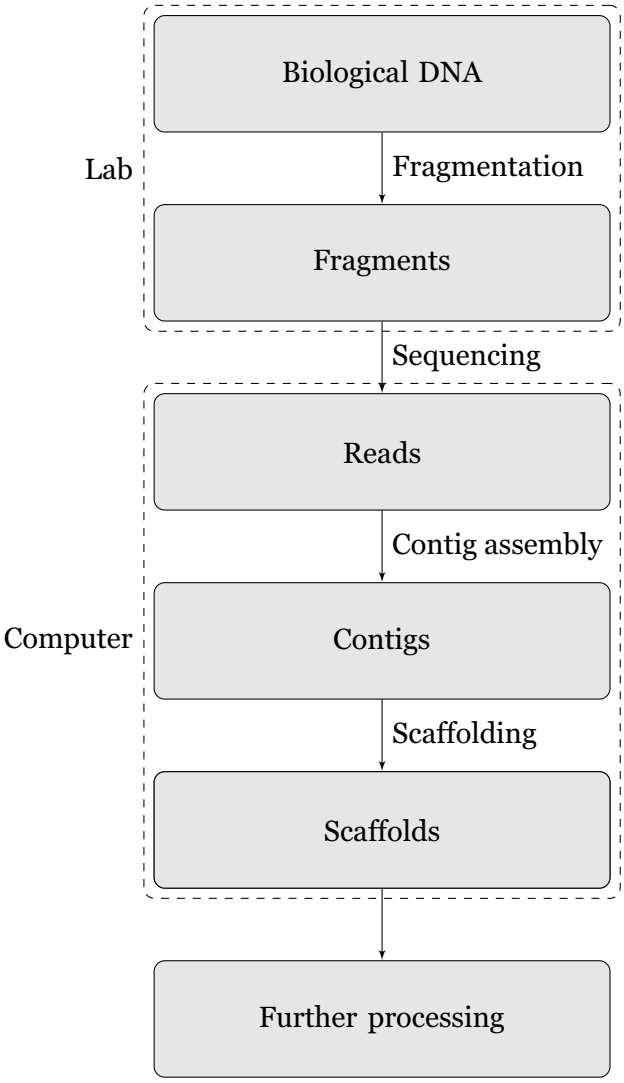

Figure 2.2: Sequencing and genome assembly

setup does not have such a reference genome available (it has not been sequenced before), and must therefore rely solely on the reads to infer the original sequence of the target.

Considering the jigsaw analogy in section 2.3 on page 7, *de novo* assembly can be equivalent to solving a jigsaw puzzle without having the resulting image available as a guide. Solving such a puzzle therefore rely on similarities of patterns and shapes between the pieces to solve the puzzle.

This thesis is mainly concerned with *de novo* assembly. This is a much harder problem than comparative assembly, and is still not considered to be completely solved [24]. The comparative approach is still worth mentioning, as the proposed method utilizes concepts from both approaches.

In the further discussion, the term *genome assembly* is used to depict the entire process of assembling a genome, including creating contigs and scaffolds (see section 2.3.2 on page 11 and section 2.3.3 on page 12). Separately, these two procedures are referred to as *contig assembly* and *scaffolding*.

2.3.1 Sequencing

Sequencing is the process of producing reads from biological material, as depicted in fig. 2.2.

Sequencing machines

The first sequencing technique was developed by Frederick Sanger in 1977 [24]. His method was able to determine sequences from 15 to 200 bases [28]. A selection of sequencing techniques is shown in table 2.2.

Technique	Read lengths
Roche/454	400 bp
SOLiD	75 bp / 50 + 25 bp
Illumina Hi-Seq 2000	150 bp / 100 + 100 bp
PacBio	> 1,000 bp

Table 2.2: A selection of sequencing techniques, from Table 1 in [19]

A common sequencing approach used in many techniques is called *shotgun sequencing*. This involves a random shearing of the DNA-material into smaller fragments. If only one molecule of DNA is used, it is impossible to assemble these fragments. This is solved (to some extent) by replicating the DNA several times. This increases the possibility of fragments covering the entire genome, and with sufficient overlap making it possible to puzzle them together. The number of replications is known as *coverage*.

A recent trend is to use sequencing technologies which produces *very short* reads, since these technologies are quick and inexpensive [24].

```
AACGCTAGGTCCGGCTAGGTC(...)TCTCCGCGGGCTAGGTCCGACG <-- Biological DNA
AACGCTAGGTCCGGC                 CGGGCTAGGTCCGACG
-> Read 1.1 -->                 <--  Read 1.2 <-
             |-- Insert size --|
```

Listing 2.3: Visualization mate-pairs and paired-end reads

Paired-end reads and mate-pairs

A common technique used in many sequencing machines is called *paired-end reads*. Longer fragments of DNA are read from both ends with an *insert size*. This insert size is the estimated number of nucleotides between the two ends.

A mate-pair is similar to a paired-end read, only with a much larger insert size, which can be estimated within a given distribution. Such insert sizes can be several thousand bases.

The use of paired-end reads and mate-pairs can help to solve the problem of *repeats* in the genome, which are almost identical sequences occuring more than once within the genome. Mate-pairs are also useful in the process of scaffolding (section 2.3.3 on page 12). Listing 2.3 shows an example of how both mate-pairs and paired-end reads may look.

Sequencing errors and correction

Different sequencing machines leads to different types of errors. Errors in the base-calling[3] can produce reads which could be identified as overlapping by the assembler program, but are missed due to these errors [17].

Some sequencing technologies have a non-uniform error-rate throughout the reads they produce, ranging from 0.3% at the beginning to 3.8% at the end of reads [10].

One remedy for errors in reads is to count and mask k-mers [17]. k-mers are unique DNA sequences of length k.

During sequencing, an estimated *coverage* is calculated, which suggests that the number of distinct k-mers present in the reads should be within a distribution around this coverage value — especially for a large k [17].

If a k-mer is found to be present in the reads only once or twice, it most likely comes from a sequencing error [17].

Reads containing such low-coverage k-mers can thus be removed from the set of reads before further processing. True k-mers are assumed to be distributed around the calculated coverage of the sequencing project while k-mers outside this distribution are marked as suspicious and/or removed from the dataset.

[3] Base-calling is the determination of a base by the sequencing machine

2.3.2 Contig assembly

This section describes the process of creating contiguous[4] sequences of DNA from the reads produced by a sequencing machine, as described in section 2.3.1 on page 9. Only a *de novo* setting is considered.

As mentioned, *de novo* assembly must exploit the information in the reads to restore parts of — or the entire — target genome. An assembly algorithm is not likely to produce *one* complete sequence covering the entire target genome. Instead the assembler will produce several contiguous sequences called *contigs*.

Considering the jigsaw analogy from section 2.3 on page 7, the process of creating contigs is equivalent to linking certain pieces from the jigsaw together into groups. There can be sufficient information available to link the pieces together inside a group, but the placement and orientation of the groups in the entire image remains unknown.

A contig is a contiguous sequence, believed to reside somewhere in the target genome. Such a contig can be from either of the two strands in the target, and in any position, making the set of contigs unoriented and unordered.

Contigs are assembled from reads in different ways. There are three major approaches: Greedy, Overlap-Layout-Consensus, and Eulerian-based. Based on the overview article *Genome assembly reborn* by Mihai Pop [24], the different approaches are explained below.

Greedy

This approach utilizes overlaps between reads in a *greedy* way, by always considering what the best next step is, choose it, and never look back to reconsider previous choices.

Overlap is defined as "[when] the prefix of one of the reads shares sufficient similarity with the suffix of another read" [24].

Greedy assemblers start by choosing an unassembled read. Then this read is merged with any other unassembled read found, meeting a defined criterion. This criterion can be the length of overlap and/or the percentage of matching bases between the reads. This continues until no more reads matches the criterion.

Overlap-Layout-Consensus

The main idea of an Overlap-Layout-Consensus (OLC) algorithm is to find all overlap between reads, and *merge* them together to create longer sequences from multiple reads.

Ideally, there would be a sufficient amount of overlaps making it possible to merge all the reads into one contiguous sequence, as visualized in listing 2.4. The example shows perfect overlaps with all bases in the overlapping regions matching perfectly.

[4] in the sense "being in actual contact", without gaps [21]

```
Read 1     ACGGGCGAGCGGCGAGC
Read 2           AGCGGCGAGCTCGAGCGACG
Read 3                CGAGCTCGAGCGACGGGACGTTG
Read 4                           GACGGGACGTTGAGCGAGCGG
Read 5                                  CGTTGAGCGAGCGGGGA
Read 6                                              CGGGGACGGTTGCATG
Consensus ACGGGCGAGCGGCGAGCTCGAGCGACGGGACGTTGAGCGAGCGGGGACGGTTGCATG
```

Listing 2.4: Example of a simple alignment of reads

OLC-based assemblers have an initial step called the *overlapper*, which finds such overlaps. This is a time-consuming operation, as it requires all reads to be compared to each other — $\binom{n}{2}$ for n reads [24]. This produces an *overlap graph*: A graph where the reads are represented by vertices, and edges between the vertices represents overlap as calculated in the initial step.

The next step is the *layout*, where the "ultimate goal is a single path that traverses each node in the overlap graph exactly once" [24]. This is equivalent to the well-known graph problem of *Hamiltonicity*, which has been proven to be **NP**-complete [2].

Paths found in the layout-step are used to create sequences based on reads the paths traverses in the final step, called the *consensus*.

Celera [22] is an OLC-based assembler used in the first assembly of the human genome in 2001.

Eulerian

Both greedy and OLC-based assembly approaches scale poorly — a large increase in the number of reads will proportionally increase the assembly complexity. In addition, the **NP**-completeness of finding Hamiltonian paths makes it hard to solve in a straight-forward manner. Eulerian assemblers mitigates this by creating a graph where the goal is to find *Eulerian* paths instead of Hamiltonian paths. A Eulerian path includes every edge in the graph exactly once [15].

The graph created is a *de Bruijn* graph, named after the Dutch mathematician Nicolaas de Bruijn. In a *de Bruijn* graph vertices are created for each distinct k-mer present in the reads (as described in section 2.3.1 on page 10). Two vertices are connected if they overlap by $k-1$ nucleotides [33].

In contrast to the **NP**-completeness of Hamiltonicity, Eulerian paths can be found in polynomial time [2], proportional to the number of edges [7].

Velvet [33] is an Eulerian assembly algorithm based on *de Bruijn* graphs and is used to create the contigs used in this thesis.

2.3.3 Scaffolding

As mentioned in section 2.3.2, the set of contigs produced by an assembler is unoriented and unordered: The placement of each contig and their relative orientation is unknown. The process of determining the relative

order and relative orientation of all or some of the contigs is called *scaffolding*.

A scaffold is a non-contiguous sequence based on linked contigs. If two contigs are believed to be adjacent in the target with an inter-contig distance, a scaffold can be created by inserting a number of N-symbols between the two contigs, representing a sequence with unknown nucleotides. This number corresponds to the assumed distance, called a *gap estimate*. The N-symbol means any of the 4 nucleotides (see table 2.3 on page 15).

If the contigs are determined to be oppositely oriented, one of them is converted to its *reverse complement*.

Scaffolding software

Many assemblers, including Velvet, are able to create scaffolds of the contigs produced. In addition to this there are standalone scaffolding programs such as SSPACE [3], BAMBUS [25], MIP Scaffolder [26] and OPERA [12]. These programs uses different approaches to the scaffolding program with mate-pair information.

Other programs such as GRASS [14] and ABACAS [1] are able to use additional information from a reference genome in the scaffolding process.

Mate-pairs in scaffolding

As mentioned in section 2.3.1 on page 10, mate-pairs can be used in a scaffolding algorithm. This is done by mapping each end of a mate-pair to the contigs produced. If each end maps to separate contigs c_1 and c_2 (and only these contigs), this suggests that c_1 and c_2 resides in the target-genome with a distance within the insert-size distribution of the mate-pairs under consideration. If several mate-pairs unambiguously maps to the same contigs, this strengthens the belief that they are indeed connected.

2.4 Sequence alignment

2.4.1 Nucleotide level

Considering the types of mutations explained in section 2.1.5 on page 6, a single insertion could be sufficient to cause the (mutated) DNA sequence to look entirely different from the original sequence, when only comparing nucleotides pairwise at each position in the genome. Therefore, methods have been developed to find the *optimal* alignments of sequences, where insertions, deletions and mutations are taken into account.

Dynamic programming is a precise technique to create optimal alignments of two or more sequences. This can be done either locally with the Smith-Waterman algorithm or globally with the Needleman–Wunsch algorithm [11, 32].

Other techniques, such as BLAST [31], use heuristics for faster, but less precise, alignments.

```
>NC_010079 Staphylococcus aureus
ACTACTGCTCAAGAAATACACGATGCGAGCAATCAAATTTCAT
AACATCACCATGAATGTTCGAACACCTTATACAGTTCTTATAC
ATACTTTATAAACACTAACAGATACTCTATAGAAGGAAAAGTT
ATCCACTTATGCAATTAGAAATTACACACAAAGTTATACTATT
TTTAGCAACATATGAAAAAGTATAATTGTGTGGATAAGTCGTC
>gi|161510924|ref|NC_010063.1| Staphylococcus aureus (..)
ATTTAAAGTGCTATAATAAACTTAATATAACAAAAACCCCAAC
AAAACGTTTARAAACTGTTTATTTATATAATTTTCGATACTTG
CAAGCACCTAAATCATTAAATAAATGTAAAGCTAAATAAAAAC
TATGGAAAGAGGTTTTTTGTCATGCAAAATCAATATTTTACAG
ATCAAATCCCTAAAGTATTCTTTACCAGTGAAAATTATAAAAA
```

Listing 2.5: Example of FASTA-formatted file

```
>Scaffold1
AAAACGTTTARAAACTGTTTATTTATATAATTTTCGATACTTG
CAAGCACCTAAATCATTAAATAAATGTAAAGCTAAATAAAAAC
TATGGAAAGAGGTTTTTTGTCATGCAAAATCAATATTTTACAG
CCTAAAGTATNNNNNNNNNNNNNNNNNNNAAAAGCCTA
ATTTAAAGTGCTATAATAAACTTAATATAACAAAAACCCCAAC
AAAACGTTTARAAACTGTTTATTTATATAATTTTCGATACTTG
TATGGAAAGAGGTTTTTTGTCATGCAAAATCAATATTTTACAG
ATCAAATCCCTAAAGTATTCTTTACCAGTGAAAATTATAAAAA
```

Listing 2.6: Example of scaffold file with a gap of 20 nucleotides

2.4.2 Protein level

The same techniques can be applied when comparing the amino acids in proteins. If the sequence alignment are used to infer relatedness between two sequences, scoring matrices called BLOSUM are commonly used. They are created from multiple sequence alignments between sequences with known *homology*. This means that they share a common ancestor. These scoring matrices reflects the likelihood of different kinds of amino acid substitutions, and provides different scores for different kinds of substitutions when comparing amino acid sequences.

2.5 Data formats

This section describes two common data formats used in bioinformatics and genome assembly.

2.5.1 FASTA

FASTA is a common format for representing genomic sequences. It has one or more headers[5] containing information about the sequence (e.g. a chromosome), followed by lines of sequence data (using the 4 bases as letters) [32]. An example excerpt from a FASTA-formatted file is shown in listing 2.5.

In addition to {A, C, T, G}, there is a number of symbols representing "one of two (or more) bases". This is a way of expressing uncertainty. M

[5] If it contains more than 1 header, it is commonly referred to as a "multi-FASTA"-file

means for instance “either A or C”. N can be any nucleotide. All these codes are shown in table 2.3.

A genome is represented in a multi-FASTA file with a header for each chromosome (or plasmid), followed by their corresponding DNA sequence.

Nucleotide(s)	Code
A	A
C	C
G	G
T	T
U	U
A,C	M
A,G	R
A,T	W
C,G	S
C,T	Y
G,T	K
A,C,G	V
A,C,T	H
A,G,T	D
C,G,T	B
A,C,G,T	N

Table 2.3: Letter codes for nucleotides in FASTA files [30], using UIPAC symbols.

In this thesis, the FASTA-format is used to represent contigs, scaffolds and entire genomes.

2.5.2 FASTQ

FASTQ is a common format for storing read data from sequencing machines. In addition to representing the base-calling for each read, an associated *quality score* can be set for each of the bases [6].

The quality score is called PHRED score, and has become a *de facto* standard for such quality scores in sequencing [6]. The estimated probability of error in each single base is calculated as

$$Q_{PHRED} = -10 \times \log_{10}(P_e) \tag{2.1}$$

The quality scores are stored as single bytes, and within the range of ASCII values 64–126 with an offset[6] of 64. This gives a range of PHRED-values from 0 to 62.

Each read is represented in the following manner:

[6]For Illumina only. There are other variants, but they are not relevant for this thesis, as all the sequencing data considered are created by Illumina technologies.

```
@SRR001665.1 071112_SLXA-EAS1_s_4:1:1:672:654 length=62
GCTACGGAATAAAACCAGGAACAACAGACCCAGCACATTAACAACAAAGGGTAAAAGGCAT
+SRR001665.1 071112_SLXA-EAS1_s_4:1:1:672:654 length=62
IIIIIIIIIIIIIIIIIIIIIIIIIIIEII9IIIEIIIIIIIIIIIIII4IIIIIGIHIIIIII
@SRR001665.2 071112_SLXA-EAS1_s_4:1:1:657:649 length=62
GCAGAAAATGGGAGTGAAAATCTCCGATGAGCAGCTTGATGCGACGACGCACCTCGTTGTT
+SRR001665.2 071112_SLXA-EAS1_s_4:1:1:657:649 length=62
IIIIIIIIIIIIIIIIIIIIIIIIII8II=II;IIIIIIIIIIIIIIIIIIIIIIIIIIII
@SRR001665.3 071112_SLXA-EAS1_s_4:1:1:708:653 length=62
GAGAGAGCAGTGGGCGAGGTTGGGACATGTCATGATCTGTGGATAACATGGTGTAAGATCC
+SRR001665.3 071112_SLXA-EAS1_s_4:1:1:708:653 length=62
IIIIIIIIIIIIIIIIIIIIIII?I=IIDIIIIIIIIIIIIIIIIIIIIIIIIII4III
@SRR001665.4 071112_SLXA-EAS1_s_4:1:1:675:644 length=62
GAACATTTATTATAATCCTATTCAATTATAATAATCTACTTTTATATGCAAGACCAAATTT
+SRR001665.4 071112_SLXA-EAS1_s_4:1:1:675:644 length=62
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIHIII
@SRR001665.5 071112_SLXA-EAS1_s_4:1:1:721:668 length=62
GCTGTAGATCTGGAAATCGCAACGGAGGAAGAAAGAAAGCATAACATCAAACAAACAAATA
+SRR001665.5 071112_SLXA-EAS1_s_4:1:1:721:668 length=62
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

Listing 2.7: Example of FASTQ-file

1. Title line, starting with a `@`
2. Sequence line(s)
3. Optional repetition of title line, starting with a +
4. Quality line(s), same length as sequence line(s)

An excerpt of a FASTQ-file is shown in listing 2.7.

Chapter 3

Methods

This chapter will cover all the details of *GuideScaff*, the proposed method for guided scaffolding using related genomes.

Section 3.1 gives an overview of the method, followed by a more thorough explanation in section 3.2.

3.1 Overview

The main concept of *GuideScaff* is to align contigs to guiding genomes, and then to create links of contigs with distance estimates between them and relative orientations based on the alignments in the guiding genomes.

The underlying hypothesis is that sequence similarities between related guiding genomes may also be shared by the target genome. If this is the case, *agreeing* guiding genomes may be used to determine correct contig orientations, ordering, and distances between contigs.

Two guiding genomes are here defined as agreeing on a contig link (c_1, c_2) if both contigs are aligned to both guiding genomes in the same *relative* order, orientations and within approximately the same distance.

By relative orientation and relative order it is here meant that wherever two contigs (c_1, c_2) are aligned to a guiding genome in a specific order and in a certain orientation, the same contigs may be aligned in the opposite order (c_2, c_1) and with opposite orientations in a different guiding genome. By looking at the order and orientation in this relative way, these alignments will be considered equal in terms of agreeing on the contig link (c_1, c_2).

3.1.1 Hypothesis on optimal contig end length

When comparing related genomes, single genes are more conserved than longer sequences of DNA [5]. If contigs are correctly assembled, each contig *does* reside somewhere in the target genome, but in an unknown position.

If the contig is long, it may contain several genes. A complete match for this contig in a guiding genome will therefore mean that all the genes in the contig are present also in the guiding genome and in the same relative order. However, if the last gene in contig A aligns unambiguously in all guiding genomes, and the first gene in contig B does the same, and the

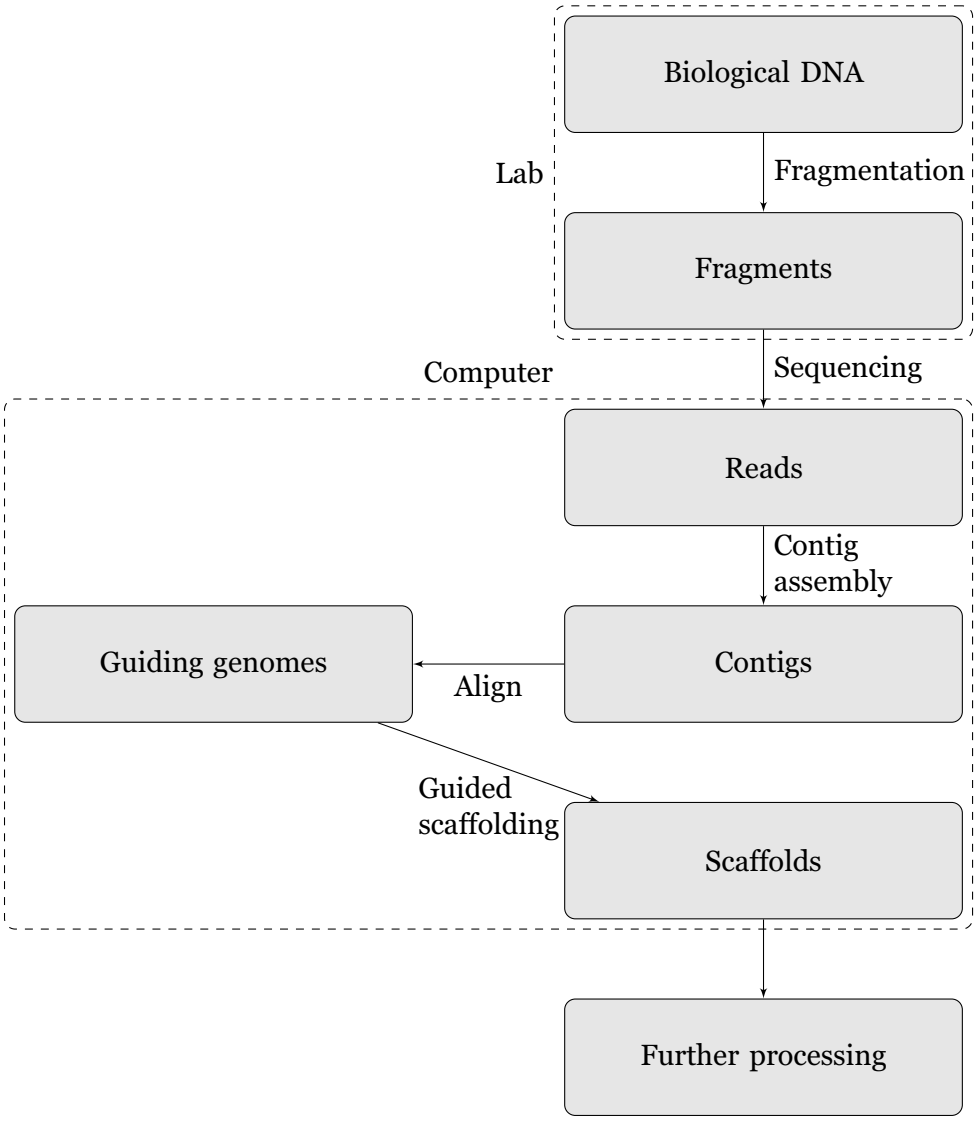

Figure 3.1: Main flow of a genome assembly process with guiding genomes

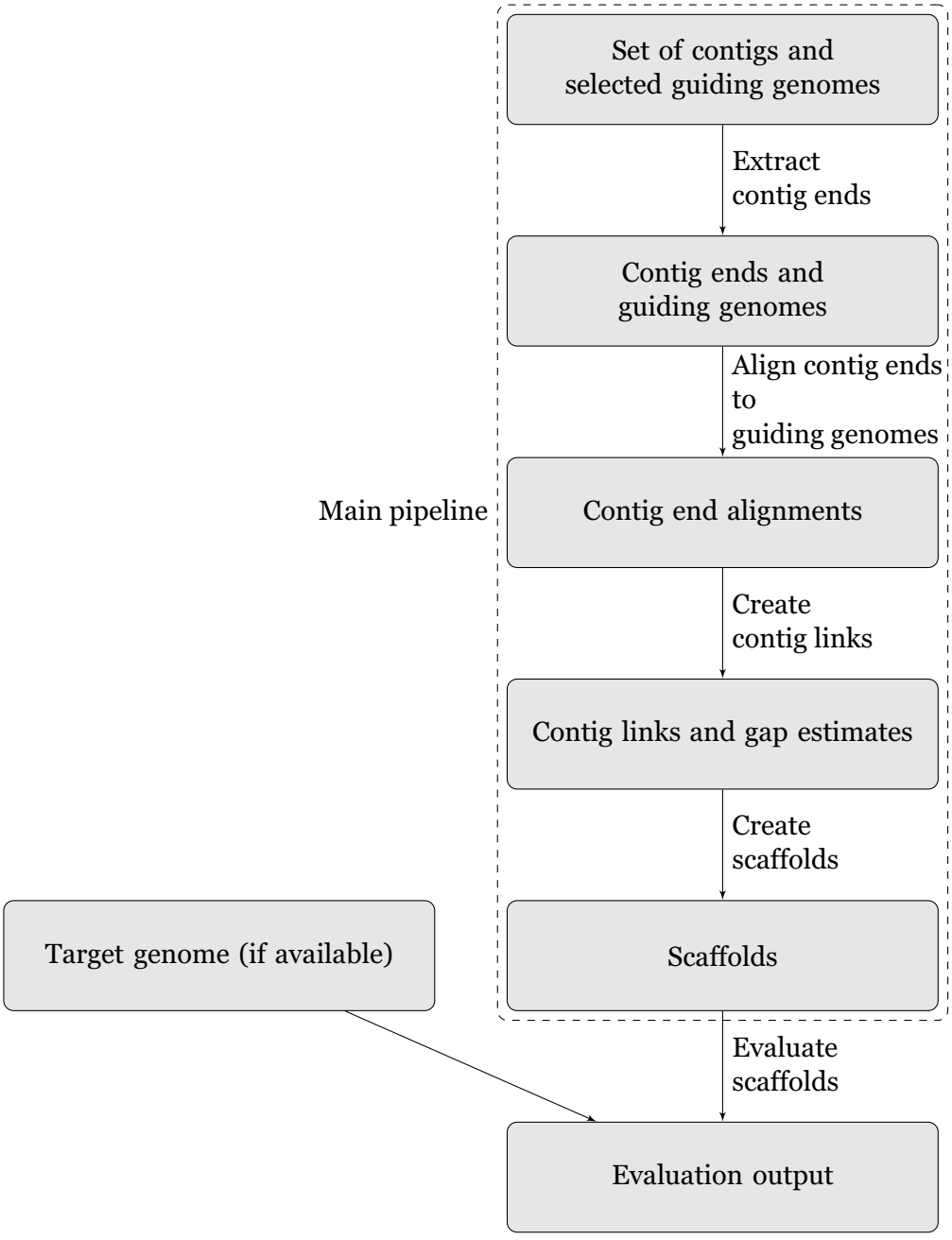

Figure 3.2: Schematic overview of *GuideScaff*

distance between these is roughly the same in each guiding genome, this could be a strong evidence of contig A and B being linked at approximately the same distance in the target genome. If this is the case all other genes inside each of these contigs are unimportant in this linking, given the assumption of an error-free contig.

In order to test this hypothesis, genes should somehow be extracted from the left end and right end of each contig.

However, genes are not considered when producing contigs. Genes can be split into different contigs, so even if a start codon is found in the beginning of a contig it is hard to determine if this is actually the start of a gene. Therefore, a simplification is done in the use of contig ends, which is to determine *one length* to use on all contigs when extracting the ends. In order to mimic the idea of choosing the first and last gene in each contig, a length approximately the average length of bacterial genes may be appropriate.

Merino *et al.* [20] reported average gene sizes for different genera[1] for both prokaryotes and eukaryotes. The average sizes reported spanned from 794 to 1,621.

If the hypothesis of gene conservation holds, an optimal contig end length could be a value around these average sizes.

3.1.2 Nucleotide- and protein alignment in guiding genomes

In order to make *GuideScaff* capable of handling guiding genomes with varying relatedness to the target, a choice of alignment type is made separately for each guiding genome.

The contigs are first aligned to a guiding genome on a nucleotide level. Then an average identity score is calculated from this alignment. If this value is below a fixed threshold, the contigs are re-aligned at a protein level.

3.2 Description of *GuideScaff*

A schematic overview of the proposed method is shown in fig. 3.2. In a true *de novo* setting, the two boxes outside the main pipeline will not apply, as the evaluation methods implemented requires the actual target sequence in order to assess the quality of the results.

Some of the metrics in the evaluation module could be used also in a true *de novo* assembly, but as the datasets used in this thesis includes the sequences of all four target genomes, this module is designed to overstep the *de novo* limitations and be as thorough as possible in the evaluation.

3.2.1 Choosing guiding genomes

The choice of guiding genomes is mainly limited by the existence and availability of related genomes already sequenced.

[1] Plural of genus [21]

One way of finding suitable guiding genomes is to use one or several of the contigs under consideration as queries to a large genomic database, and select the best matches found.

In this thesis the largest contig from each dataset was used as a query in BLAST [34], and 10 fully assembled genomes from the top results were selected as guiding genomes. This was done to find suitable guiding genomes easily. It was also done in order to find guiding genomes from other species where there were no closely related genomes available.

3.2.2 Contig preparation and contig end extraction

The contigs may be produced by any genomic assembler. All contigs used in this thesis were produced by Velvet [33] from Illumina reads. The reads for S. aureus and R. sphaeroides were error-corrected with Quake [17] prior to creating the assembly [27].

If contig end extraction is used, a contig end length N is chosen. Then each contig is either transformed to a pair of contig ends or the entire contig is kept intact.

For each contig processed, the contig length l and the contig end length N determines what:

$l > 2N$**:** A left-end and a right-end of size N are extracted from the contig.

$l = 2N$**:** The contig is split in two ends of size N.

$l < 2N$**:** The contig is kept intact.

This allows all contigs to be kept for further processing, regardless of their sizes.

In order to make the further discussion simple, “contig ends” are used to indicate both actual contig ends extracted at this stage and the entire contigs when no contig end extraction is used or when contigs are too short to have their ends extracted. This is done to differentiate the contigs *aligned* and the elements which are finally linked together and used to build scaffolds.

3.2.3 Aligning contig ends to guiding genomes

When a set of guiding genomes is chosen and contig ends are extracted, each contig end is aligned to the guiding genomes using tools from MUMmer. MUMmer is “an open source software package for the rapid alignment of very large DNA and amino acid sequences” [18]. Two tools from MUMmer are used to align on nucleotide- and protein levels: `nucmer` and `promer`, respectively. Both tools uses suffix trees to align *queries* to *references*. Suffix trees are efficient data structures which allow searching for sub-strings in linear time and space [2, 18].

Nucleotide alignment with `nucmer` is run first to give a preliminary alignment of the contig ends. A measure of average similarity score of this alignment is calculated, and this value is compared to a fixed threshold

```
$ show-tiling out.delta
>gi|49175990|ref|NC_000913.2| 4639675 bases
3551    5050    8261    1500    100.00  100.00  +   LFT_SC_0_+
13312   14811   12290   1500    100.00  100.00  +   RGT_SC_0_+
27102   28601   3381    1500    100.00  100.00  +   LFT_SC_1_+
31983   33482   6999    1500    100.00  100.00  +   RGT_SC_1_+
40482   41981   13410   1500    100.00  100.00  +   LFT_SC_2_+
55392   56891   19877   1500    100.00  100.00  +   RGT_SC_2_+
76769   78268   3228    1500    100.00  100.00  -   RGT_SC_3_-
81497   82996   10177   1500    100.00  100.00  -   LFT_SC_3_-
93174   94673   6578    1500    100.00  100.00  -   RGT_SC_4_-
101252  102751  11260   1500    100.00  100.00  -   LFT_SC_4_-
(...)
```

Listing 3.1: Example output from `show-tiling`

which determines if this alignment is satisfactory or if `promer` should be used to align the contig ends on a protein level.

After the alignment is finished, MUMmer's tool `show-tiling` is used to construct a *tiling path* of the contigs. This is a list of contig end placement in the guiding genome where each contig end is used at most once (not present if mapping quality was poor or if it mapped several places and caused ambiguity).

An example output of a tiling file is shown in listing 3.1. This listing shows an excerpt of the tiling produced when *simulated* contig ends have been aligned to the target genome.

The simulated contigs have names corresponding to their true order in the target genome, as well as a +/− symbol indicating whether they were extracted from the target as they were (+) or transformed to their reverse complements (−).

3.2.4 Contig links creation

After aligning each contig end to the guiding genomes, the tiling files created are further processed in the most essential module of *GuideScaff*, which builds contig links based on all the alignments produced.

Figure 3.3 shows a schematic overview of this module.

Contig links are created as paths of contig ends where each end of a contig must be present, unless the entire contig is present, in order to be used further. This process is similar to the greedy approach of contig assembly mentioned in section 2.3.2 on page 11, as it always considers the best *next step*.

The creation of contig links is controlled by two parameters:

- Window size w
- Minimum number of agreeing guiding genomes t

The window size w determines the size of a sliding window. This window is used when creating entries in a distance matrix M, based on the tiling files. This value must be at least 2, in which case only two-by-two contig ends are inserted as matrix entries.

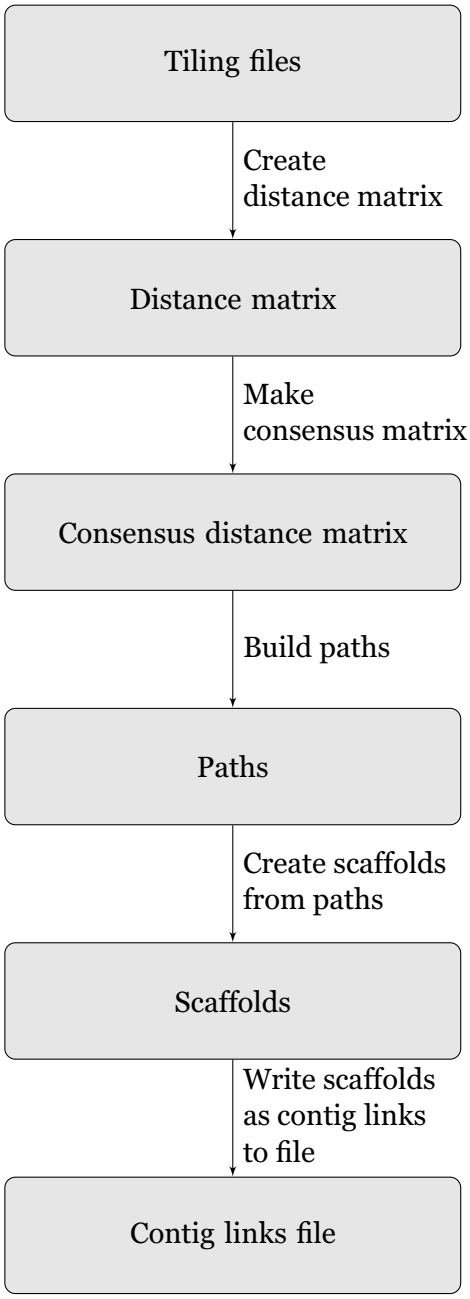

Figure 3.3: Schematic overview of contig linking module in *GuideScaff*

The minimum number of agreeing guiding genomes t is used when creating a consensus distance matrix M^c. t may range from 1 (one of the guiding genomes) to the number of guiding genomes g used. For a contig pair (c_1, c_2), $M^c_{c_1,c_2}$ will be set if at least t of the guiding genomes have c_1 and c_2 within w entries in their tiling lists. If g is used, all guiding genomes have to agree on a link.

Creating the distance matrix

A distance matrix M is created as a 2-dimensional table of contig ends and the reported distance(s) between them from the tiling files.

A matrix entry consists of

- Contig name
- Contig end indicator
- Orientation

The end indicator tells which end of the contig it was extracted from, or if the contig was used in its full length.

M is created by traversing each tiling file as depicted in the following pseudocode:

```
M ← empty matrix
for each tiling file in tiling files do
  for each chromosome in tiling file do
    window ← sliding window of size w inside chromosome
    for each pair (c_i, c_j) in window do
      M_{i,j} ← distance(c_i, c_j)
    end for
  end for
end for
```

A contig-link (c_1, c_2) within a window will then create an entry M_{c_1,c_2} with the calculated distance between c_1 and c_2. An additional entry $M_{c_2',c_1'}$ is also made, where c_1' and c_2' are from the opposite strand than c_1 and c_2. This makes a contig link (c_1, c_2) in the normal orientation equal to (c_2, c_1) in the opposite orientation.

Creating the consensus distance matrix

When a distance matrix M is created, all entries $M_{i,j}$ are lists with 1 or more distances as reported from the guiding genome alignments. A *consensus distance matrix* M^c is created in the following way:

The minimum number of agreeing guiding genomes t is used to control which matrix entries are kept in M^c by keeping entries with at least t distances: $|M_{i,j}| \geq t$.

In order to apply these distances to the scaffolding, all the t-filtered distance lists are transformed to *one* value using the *median* of all distances.

Creating paths of contig ends

Paths of contig ends are created before they are turned into scaffolds.

As long as there are unprocessed contigs, paths are grown from a contig end in the set of contig ends not yet processed. This path is further grown from its initial contig end by first adding its opposite end. Then the last appended contig end c_l is looked up in M^c. From the entries in $M^c_{c_l}$, the closest contig end is chosen, if it is not an already processed contig.

This process is repeated until all contigs are processed or if there is no way to further expand the current path.

The important concept of this algorithm is the *skip* from one end of a contig to the other. If a path starts with the left end of a large contig c_1, it is not necessary to look up this contig in M^c, because the right end of c_1 must be the next contig end in order to use the entire contig. In this lies the possible advantage of using contig ends instead of entire contigs.

This procedure results in multiple paths where contigs are present with both ends in exactly one of the directions or not at all. If no contig end alignments were found or if the agreement threshold value is set too high, this step could result in no paths.

Creating scaffolds from paths

After the creation of contig end paths, scaffolds are created by collapsing contig ends from the same contig into single contigs and using gap estimates between consecutive resulting contig links. A gap estimate between contigs i and j in the scaffold is extracted from the consensus distance matrix, $M^c_{i,j}$.

The scaffolds are written to file in an intermediate format. This allows the resulting contig links to be evaluated regardless of the actual sequences in the scaffold file produced.

The output of the contig linking is a file similar to the tiling files, only with 3 columns: gap estimate, orientation and contig name. As with the tiling files, the gap estimate in one line refers to the estimated distance between the contig in the same line and the contig in the line below.

An excerpt of an example file with contig links is shown in listing 3.2.

3.2.5 Creating final scaffold file from contig links

In this step a scaffold file is created, based on the contig links from the previous step, and the unused contigs.

Each scaffold in the contig links file is transformed into a FASTA-formatted scaffold by

- Looking up all the contigs in the scaffold in the contigs file
- Transforming the contigs marked with – into their reverse complements
- Merging all sequences of contig links (c_1, c_2)

```
>Scaffold1
235     -   SC_22_-_SRC=gi|49175990|ref|NC_000913.2|
-46     -   SC_23_-_SRC=gi|49175990|ref|NC_000913.2|
30      +   SC_24_+_SRC=gi|49175990|ref|NC_000913.2|
0       -   SC_38_-_SRC=gi|49175990|ref|NC_000913.2|
>Scaffold2
13      -   SC_321_-_SRC=gi|49175990|ref|NC_000913.2|
57      -   SC_322_-_SRC=gi|49175990|ref|NC_000913.2|
73      -   SC_323_-_SRC=gi|49175990|ref|NC_000913.2|
-32     +   SC_324_+_SRC=gi|49175990|ref|NC_000913.2|
0       -   SC_326_-_SRC=gi|49175990|ref|NC_000913.2|
```

Listing 3.2: Example of contig-links file. Each scaffold consist of at least two lines with gap estimate between current contig and the next in the first column, an orientation symbol in the second column and the contig name in the third column.

Merging linked contigs is done in different ways, depending on the gap estimate between them: If the gap estimate g is positive, a g N-symbols are inserted between the merged contig sequences. If g is negative an overlap is implied. The contigs are then attempted merged at the point they *actually* overlap. If there is no overlap, the two contigs are simply concatenated as if the gap estimate were 0.

After the contig links are transformed into scaffolds all the contigs *not* used in any of the scaffolds are appended. Finally all the resulting sequences are written to file in a multi-FASTA format.

An optional part of the pipeline is to run a breakpoints count for the results. This is done with `countBreakPoints.py`.

3.3 Implementation

GuideScaff consists of different modules, which are mainly written in Python. All programs are available at https://github.com/runarfu/GuideScaff.

A simplified pipeline in BASH is shown in listing 3.3.

The first module of *GuideScaff* is `makeContigEnds.py`, which extract ends from contigs with a given length `nCut`. If `nCut` is 0, contigs are kept in full lengths.

Then `runxmer.sh` uses `nucmer` to align all contig ends to the guiding genomes. If the results are insatisfactory for any given guiding genome, `promer` is run.

The main module in *GuideScaff* is `makeContigLinks.py` which analyzes any number of tiling files produces by `nucmer` or `promer` and outputs one or multiple clusters of contig-links.

Finally, `makeScaffolds.py` creates a scaffold file in FASTA format, based on the contig links.

Other programs are used to produce different statistics in the evaluation. A script used in Assemblathon 2 [4] is also run and used to calculate the N50 measure for both contigs and scaffolds.

```
#!/bin/bash

# Input variables
DIR=$1
NGUIDES=$2
WINDOWSIZE=$3
THRESHOLD=$4
NCUT=$5
GUIDE_DIR=$DIR/guidingGenomes

# Extract ends with length NCUT from contigs
makeContigEnds.py --inputFile  $DIR/contigs.fasta
                  --outputFile $DIR/contigPairs.fasta
                  --nCut $NCUT

# Align the contig ends to each of the guiding genomes
# using GNU Parallel
parallel runxmer.sh {} $DIR/contigPairs.fasta $NCUT :::
         $GUIDE_DIR/*.fasta

# Make contig links from tiling-files produced by nucmer or promer
makeContigLinks.py --inputFiles $GUIDE_DIR/*.fasta.$NCUT.tiling
                   --output     $DIR/contigLinks
                   --nGuides    $NGUIDES
                   --windowSize $WINDOWSIZE
                   --threshold  $THRESHOLD

# Use contig links to build scaffolds in FASTA format
makeScaffolds.py  --inputFile   $DIR/contigLinks
                  --outputFile  $DIR/scaffolds.fasta
                  --contigsFile $DIR/contigs.fasta
```

Listing 3.3: Pipeline for *GuideScaff* as BASH-script

```
$ parallel nucmer {} contigs.fasta ::: *.fasta
```

Listing 3.4: Example run of `nucmer` with GNU Parallel. `{}` works as a placeholder for each file with suffix `fasta`, and separate instances of `nucmer` are started simultanously.

3.3.1 Additional speed gain with GNU Parallel

Though the alignments are created quickly, additional speed is gained by using GNU Parallel [29] to start concurrent alignment processes for each guiding genome. Listing 3.4 shows how GNU Parallel can be used to align the contigs in `contigs.fasta` to all FASTA-files in the current directory.

An example run of `nucmer` with 481 contigs aligned to 20 different guiding genomes ran on an Intel i7 desktop computer with 4 cores in 173 seconds when run sequential. Using GNU Parallel, it ran in 38 seconds.

Chapter 4

Evaluation

The evaluation of a genome assembly can be done in many ways. An important difference in how an assembly may be evaluated is whether the actual target genome is available.

All the data sets in this thesis includes the known target sequences. Therefore the further discussion will focus on evaluation metrics used when such a "golden truth" sequence is known.

In the evaluation of genome assemblies there is a difference between inter-contig errors and scaffolding errors. As this thesis focus on scaffolding, inter-contig errors will not be considered. The assembler used to create the contigs are assumed to be perfect, having produced error-free contigs, or with such low error-rates that the further investigation is not biased by possible errors within the contigs.

4.1 Commonly used evaluation metrics

4.1.1 N50

The N50 value is a widely used metric for evaluating both assemblies and scaffolds. It is defined as "the size of the smallest contig (or scaffold) such that 50% of the genome is contained in contigs of size N50 or larger" [14].

A large N50 value for the produced scaffolds means that the scaffolds are long, which in turn means that many contigs are linked together.

A limitation of this metric is that it is *only* a measure of contig/scaffold sizes. This means that a scaffold containing a concatenation of all the contigs from an assembler would give a large N50 value. However, the relative placement of the contigs inside this scaffold could be wrong, as well as their relative orientations and the estimated gap sizes between the contigs. The N50 is still a useful metric, but other evaluation metrics are also needed to fully assess the *quality* of the scaffolds produced.

4.1.2 Breakpoints

Gritsenko *et al.* uses a concept called *breakpoints* to assess scaffold qualities.

Breakpoints are counted by considering contig links in the scaffolds as pairs of contigs (c_1, c_2). A breakpoint is counted for such a contig pair if *at least one* of the following hold:

1. **Contigs maps to different chromosomes in the target genome.** This means that both contigs in a pair can be found in the target sequence, but in different chromosomes[1].

2. **The relative orientations of contigs inside a scaffold does not correspond to the relative orientations of the same contigs in the target.** This measure is calculated as the least of two numbers: (i) The number of contigs which are oriented different than the same contigs in the target genome, and (ii) the number of contigs oriented the same way as in the target genome. If all n contigs inside a scaffold are suggested to be oriented in the normal direction and the same contigs are oriented in the opposite direction in the target genome, this results in two numbers: n and 0. n of the contigs in the scaffold are oriented differently than the same contigs in the target genome, and 0 of them are oriented in the same way as in the target genome. Since this is a measure of *relative* orientations, the least of these numbers (0) is chosen, as the relative orientation of the entire scaffold is correct, though incorrect as an *absolute* orientation.

3. **The relative order does not correspond to the relative order in the target.** This measure is similar to the relative orientations measure, but considers the relative *positions* of contigs in a scaffold and in the target genome. For each scaffold two numbers are calculated: (i) The number of pairs (c_1, c_2) within a scaffold which are differently ordered in the target genome (where c_2 comes *before* c_1) in the normal direction, and (ii) the same number for the reverse direction. If a scaffold is listed with 5 contigs as c_4, c_3, c_2, c_1, and the true order of the same contigs according to the target genome is c_1, c_2, c_3, c_4, then the numbers of relative ordering errors are 0 and 3 (3 pairs). Again, this is a *relative* order, so the least number is chosen and reported. In the aforementioned example the least number will be 0, since the order is correct if considered backwards.

4. **The gap estimate between two linked contigs are wrong, given a Δ.** This means that the gap estimate g_{estimate} must not differ from the true gap size g_{true} with more than Δ, or else it will be counted as a breakpoint:

$$\text{breakpoint}_{\text{gap}} = \begin{cases} \text{Yes}, & |g_{\text{estimate}} - g_{\text{true}}| > \Delta \\ \text{No}, & |g_{\text{estimate}} - g_{\text{true}}| \leq \Delta \end{cases} \tag{4.1}$$

The methods used in GRASS involved the scaffolds being aligned to the target genome. *GuideScaff* uses an intermediate format (section 3.2.4

[1] In different chromosomes or plasmids — these terms are here used interchangeably.

on page 25) with explicit contig links allowing breakpoints to be counted without mapping the actual scaffolds.

GRASS reported the number of breakpoints for Δ-values 500 bp and 10,000 bp. In the evaluation of *GuideScaff* Δ-values 100 bp and 1,000 bp are used in addition to these values in order to further nuance the evaluation.

4.1.3 Modifications to breakpoint counting

To a large extent, the concept of breakpoints covers the type of errors possible in genomic scaffolds. However, it gathers the 4 different errors into *one* measure. For a contig link, *one* breakpoint is reported regardless of how many of the different errors were made.

In order to assess *which* errors are made, the evaluation module of *GuideScaff* reports all types of errors individually. This makes it easier to explore which errors are more frequent and allows for experiments with modifications to *GuideScaff*, to see what remedies different kinds of errors.

Another reason for looking at each error type individually is that the different errors may not be equally severe. It could for instance be considered a gross error to link two contigs together when they actually map to different chromosomes in the target. A gap estimate between two contigs exceeding a small Δ value could for instance for instance be considered less severe.

The number of breakpoints for the different error types is also reported as a *relative* number. This is for instance the number of incorrect gap estimates divided by the number of contig pairs in the scaffolds. This allows for fair comparisons between scaffolds of various sizes.

The evaluation module of *GuideScaff* reports separate numbers of breakpoints and a relative number $\in [0, 1]$ for each of them.

4.1.4 Measure of correctness

When running *GuideScaff* on different datasets and with different parameter settings, it is hard to assess the performance with respect to *all* the individual error types. One single measure is then used to assess *correctness* of the resulting scaffolds produced.

The correctness measure c is a combination of all *relative* breakpoints counted. This is calculated as

$$c = \prod(1 - e_i), e_i \in [0, 1] \tag{4.2}$$

where e is a relative measure of errors made in the scaffolds.

This makes a rough but useful metric to use when evaluating the scaffold results. If no errors are made, all e_i will be 0, and c will be 1, meaning 100% correctness. If at least one $e_i = 1$ this cancels the entire expression, and c will be 0. This implies that *all* contig pairs map to to different chromosomes, *all* contigs are incorrect relative oriented or incorrect relative ordered, or that not a single gap estimate is within the limit Δ.

Chapter 5

Materials

In this chapter all the datasets that were used with *GuideScaff* are presented. All the data used are available at http://hyperbrowser.uio.no/dev2/static/downloads/runarfu_master_supplementaryData.zip

In this thesis four bacterial genomes were used as target genomes. All of these genomes are relatively small, ranging from about 2.8 million bases to about 4.6 million bases. They vary in the number of chromosomes and plasmids, which allows an assessment of correct chromosome/plasmid placement in the scaffolds.

The contigs and scaffolds used were produced in two projects of genome assembly: *GAGE: A critical evaluation of genome assemblies and assembly algorithms* [27] and *GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies* [14].

All these datasets came from real sequencing data which were produced using Illumina technologies. The authors of GAGE and GRASS assembled these reads to contigs and scaffolds with Velvet.

For each target genome, the following is included:

- The target sequence. This is the gapless sequence of the target genome, and is used as the "golden truth" in all assessments of the resulting scaffolds.
- A set of contigs produced with Velvet from libraries of short reads created by Illumina technologies.
- A set of scaffolds created with Velvet.

5.1 Target genomes and contigs

The four target genomes used are

- Escherichia coli str. K-12 substr. MG1655
- Pseudoxanthomonas suwonensis 11-1
- Rhodobacter sphaeroides 2.4.1
- Staphylococcus aureus subsp. aureus USA300_TCH1516

As table 5.1 shows, the target genomes vary in both size and number of chromosomes.

Each of the 4 target genomes had been sequenced with Illumina technologies, with both the original reads and the complete sequences freely available online.

The data for S. aureus and R. sphaeroides were contigs assembled from Illumina reads with Velvet [33] by Saltzberg *et al.* [27], and the data for P. suwonensis and E. coli were contigs produced by Gritsenko *et al.* , also using Illumina reads and Velvet.

All the contig sets used are shown in table 5.3. The reads which the contigs were produced from are shown in table 5.2.

Escherichia coli str. K-12 substr. MG1655 chromosome		
Chromosome 1	4,639,675	NC_000913.2
Pseudoxanthomonas suwonensis 11-1		
Chromosome 1	3,419,049	CP002446.1
Rhodobacter sphaeroides 2.4.1		
Chromosome 1	3,188,609	CP000143
Chromosome 2	943,016	CP000144
Plasmid A	114,045	DQ232586
Plasmid B	114,178	CP000145
Plasmid C	105,284	CP000146
Plasmid D	100,828	CP000147
Plasmid E	37,100	DQ232587
Staphylococcus aureus subsp. aureus USA300_TCH1516		
Chromosome 1	2,872,915	NC_010079
Plasmid pUSA01-HOU	3,125	NC_012417.1
Plasmid pUSA300HOUMR	27,041	NC_010063.1

Table 5.1: Target genomes and their chromosomes/plasmids. Size is the number of bases in each sequence.

Target genome	SRA accession
E. coli	SRR001665, SRR001666
P. suwonensis	SRR097515, SRR191848
R. sphaeroides	SRX033397, SRX016063
S. aureus	SRX007714, SRX016063

Table 5.2: Read data used with Velvet in GRASS [14] and GAGE [27] to create the contigs. All reads are from Illumina sequencing and freely available at Sequence Read Archive (SRA) [31] http://www.ncbi.nlm.nih.gov/sra

Target genome	#Contigs	#Contigs (filtered)	Maximum contig length
E. coli	481	481	73,062
P. suwonensis	303	303	90,572
R. sphaeroides	809	583	60,714
S. aureus	301	162	169,214

Table 5.3: The contig sets used. All contigs were created with Velvet [33] from real sequencing data [14, 27]. Contigs shorter than a certain threshold were removed as in GRASS and GAGE, using thresholds of 150 and 200 bp, respectively. The contigs from GAGE were already filtered.

	E. coli	P. suwonensis	R. sphaeroides	S. aureus
# Scaffolds	41	32	25	9
N50	171,726	57,614	353,027	762,333
Max scaffold size	312,219	153,169	770,958	989,718
Min scaffold size	1,425	7,801	475	7,400

Table 5.4: Scaffolds produced with Velvet using read libraries listed in table 5.2. Scaffold files were retrieved from GAGE and GRASS. The number of scaffolds is reported as the number of *actual* scaffolds, where at least two contigs are linked. Maximum and minimum sizes of the scaffolds are calculated with unlinked contigs excluded. This is done in order to compare these results to the results of *GuideScaff* later.

5.2 Guiding genomes

The goal of the proposed method (*GuideScaff*) is to link contigs from each of the targets into scaffolds, with the aid of related *guiding genomes*.

As mentioned, a guiding genome may be similar on a nucleotide level, if it is closely related. If it is more distantly related, it may be more similar on a protein level.

In order to test the performance and limitations of *GuideScaff*, 10 guiding genomes were chosen for each of the targets. Some guiding genomes were very closely related, whereas others were more distantly related.

All guiding genomes were found using BLAST, with the largest contig from each dataset as queries.

When selecting guiding genomes for S. aureus and R. sphaeroides, a BLAST match in *one* chromosome of an organism was expanded to include *all* the chromosomes and plasmids in this organism. The other two target genomes consist of single chromosomes, thus only the single chromosomes matched in BLAST were used as guiding genomes for these.

The guiding genomes used are shown in tables 5.6, 5.9, 5.7 and 5.8.

To get an idea of the sequence similarities between the target genomes and their selected guiding genomes, Mauve [8] was used to find conserved regions common in all the genomes used. Mauve calls this LCBs (Locally Collinear Blocks).

All the guiding genomes were prefixed with a number from 01 to 10. These numbers corresponded to the order in which they were reported from BLAST. In each Mauve plot the top-most sequence is the target genome, followed by the guiding genomes from the best match to the worst match according to the BLAST results.

In fig. A.1 on page 68, fig. A.2 on page 69, fig. A.3 on page 70 and fig. A.4 on page 71 these plots are shown. These plots are available in vectorized formats at http://heim.ifi.uio.no/runarfu/master/mauveAlignments/.

Target genome	Size	Contig ID
E. coli	73,062	NODE_70_length_73032_cov_38.096340
P. suwonensis	90,572	NODE_42_length_90514_cov_256.119507
R. sphaeroides	60,714	velvet.380.9 100332 161045
S. aureus	169,214	velvet.93.5 100698 269911

Table 5.5: Largest contigs in datasets used to search for guiding genomes with BLAST.

E. coli DH1		
Chromosome	4,621,430	NC_017638.1
E. coli BW2952		
Chromosome	4,578,159	NC_012759.1
E. coli str. K-12 substr. W3110		
Chromosome	4,646,332	NC_007779.1
E. coli str. K-12 substr. DH10B		
Chromosome	4,686,137	NC_010473.1
E. coli HS		
Chromosome	4,643,538	NC_009800.1
E. coli ATCC 8739		
Chromosome	4,746,218	NC_010468.1
E. coli E24377A		
Chromosome	4,979,619	NC_009801.1
E. coli O26:H11 str. 11368		
Chromosome	5,697,240	NC_013361.1
E. coli W		
Chromosome	4,897,452	NC_017664.1
E. coli KO11FL		
Chromosome	5,021,812	NC_017660.1

Table 5.6: Guiding genomes used for E. coli

Pseudoxanthomonas spadix BD-a59		
Chromosome	3,452,554	NC_016147.2
Xanthomonas campestris pv. vesicatoria str. 85-10		
Chromosome	5,178,466	NC_007508.1
Xanthomonas axonopodis pv. citrumelo F1		
Chromosome	4,967,469	NC_016010.1
Xanthomonas campestris pv. raphani 756C		
Chromosome	4,941,214	NC_017271.1
Xanthomonas axonopodis pv. citri str. 306		
Chromosome	5,175,554	NC_003919.1
Xanthomonas campestris pv. campestris str. 8004		
Chromosome	5,148,708	NC_007086.1
Xanthomonas campestris pv. campestris str. ATCC 33913		
Chromosome	5,076,188	NC_003902.1
Xanthomonas albilineans GPE PC73		
Chromosome	3,768,695	NC_013722.1
Xanthomonas campestris pv. campestris str. B100		
Chromosome	5,079,002	NC_010688.1
Stenotrophomonas maltophilia R551-3		
Chromosome	4,573,969	NC_011071.1

Table 5.7: Guiding genomes used for P. suwonensis

Rhodobacter sphaeroides ATCC 17029		
Chromosome 1	3,147,721	NC_009049.1
Chromosome 2	1,219,053	NC_009050.1
Plasmid pRSPH01	122,606	NC_009040.1
Rhodobacter sphaeroides KD131		
Chromosome 1	3,152,792	NC_011963.1
Chromosome 2	1,297,647	NC_011958.1
Plasmid pRSKD131A	157,345	NC_011962.1
Plasmid pRSKD131B	103,355	NC_011960.1
Rhodobacter sphaeroides ATCC 17025		
Chromosome	3,217,726	NC_009428.1
Plasmid pRSPA01	877,879	NC_009429.1
Plasmid pRSPA02	289,489	NC_009430.1
Plasmid pRSPA03	121,962	NC_009431.1
Plasmid pRSPA04	36,198	NC_009432.1
Plasmid pRSPA05	13,873	NC_009433.1
Polymorphum gilvum SL003B-26A1		
Chromosome	4,649,365	NC_015259.1
Plasmid pSL003B	69,598	NC_015258.1
Paracoccus denitrificans PD1222		
Chromosome 1	2,852,282	NC_008686.1
Chromosome 2	1,730,097	NC_008687.1
Plasmid 1	653,815	NC_008688.1
Ruegeria pomeroyi DSS-3		
Chromosome	4,109,442	NC_003911.11
Megaplasmid	491,611	NC_006569.1
Rhodobacter capsulatus SB 1003		
Chromosome	3,738,958	NC_014034.1
Plasmid pRCB133	132,962	NC_014035.1
Agrobacterium sp. H13-3		
Chromosome	2,823,930	NC_015183.1
Chromosome linear	2,148,289	NC_015508.1
Plasmid pAspH13-3a	601,551	NC_015184.1
Agrobacterium fabrum str. C58		
Chromosome circular	2,841,580	NC_003062.2
Chromosome linear	2,075,577	NC_003063.2
Plasmid At	542,868	NC_003064.2
Plasmid Ti	214,233	NC_003065.3
Ruegeria sp. TM1040		
Chromosome	3,200,938	NC_008044.1
Mega plasmid	821,788	NC_008043.1
Plasmid unnamed	130,973	NC_008042.1

Table 5.8: Guiding genomes used for R. sphaeroides

S. aureus subsp. aureus str. Newman		
Chromosome	2,878,897	NC_009641.1
S. aureus subsp. aureus NCTC 8325		
Chromosome	2,821,361	NC_007795.1
S. aureus subsp. aureus COL		
Chromosome	2,809,422	NC_002951.2
Plasmid pT181	4,440	NC_006629.2
S. aureus subsp. aureus USA300_FPR3757		
Chromosome	2,872,769	NC_007793.1
Plasmid pUSA03	37,136	NC_007792.1
Plasmid pUSA01	3,125	NC_007790.1
Plasmid pUSA02	4,439	NC_007791.1
S. aureus subsp. aureus T0131		
Chromosome	2,913,900	NC_017347.1
S. aureus subsp. aureus TW20		
Chromosome	3,043,210	NC_017331.1
Plasmid pTW20_2	3,011	NC_017332.1
Plasmid pTW20_1	29,585	NC_017352.1
S. aureus subsp. aureus str. JKD6008		
Chromosome	2,924,344	NC_017341.1
S. aureus subsp. aureus VC40		
Chromosome	2,692,570	NC_016912.1
S. aureus subsp. aureus ED98		
Chromosome	2,824,404	NC_013450.1
Plasmid pAVY	1,442	NC_013451.1
Plasmid pT181	4,440	NC_013452.1
Plasmid pAVX	17,256	NC_013453.1
S. aureus subsp. aureus N315		
Chromosome	2,814,816	NC_002745.2
Plasmid pN315	24,653	NC_003140.1

Table 5.9: Guiding genomes used for S. aureus

Chapter 6

Results

The following sections present the results of running *GuideScaff* on all four datasets described in chapter 5. All result data presented in this chapter are also available in http://hyperbrowser.uio.no/dev2/static/downloads/runarfu_master_supplementaryData.zip.

For each run, the resulting scaffolds are reported in terms of the following evaluation measures:

- N50 value of the scaffolds
- Number of contigs scaffolded[1]
- Number of contig links incorrectly placed in the same chromosome
- Number of contigs incorrectly ordered
- Number of contigs incorrectly oriented
- Gap estimate errors exceeding 500 bp
- Gap estimate errors exceeding 10,000 bp

6.1 Initial runs of *GuideScaff* on all datasets

In order to establish a baseline of performance, the pipeline was run on all 4 datasets with *one* guiding genome from each dataset. This guiding genome was chosen to be the first match in the BLAST results, as discussed in section 3.2.1 on page 20.

6.1.1 Using entire contigs

Contig ends were first *not* extracted. Parameters were fixed at $w = 2$, $t = 1$, meaning that the tiling files were processed with pairs of contigs, and only one guiding genome had to agree on contig links. The results that this setup produced are shown in table 6.1 on page 44.

[1] The number of contigs *used* in one of the scaffolds

E. coli

With E. coli, 421 of 481 contigs were gathered in 4 scaffolds. All of these contigs were oriented and ordered correctly within each scaffold. At 500 bp resolution 15 contig links had wrong gap estimates, and at 10,000 bp resolution 2 of the gap estimates were wrong.

P. suwonensis

With P. suwonensis, 97 of 303 contigs were gathered in 3 scaffolds. 37 contigs were incorrectly oriented, and 38 were incorrectly ordered within the scaffolds they were placed in. 77 of 94 contig links had wrong gap estimates at 500 bp resolution, and 61 gap estimates were wrong at 10,000 bp resolution.

R. sphaeroides

R. sphaeroides consists of 2 chromosomes and 5 plasmids. Its first guiding genome consists of 2 chromosomes and 1 plasmid. With this genome as guiding genome, 387 of 583 contigs were gathered in 13 scaffolds. One chromosomal error was made, meaning that one contig pair mapped to different chromosomes in the target genome. 3 contigs were incorrectly oriented, and 2 contigs were incorrectly ordered. 36 of 374 contig links had wrong gap estimates at 500 bp resolution, and 18 gap estimates were wrong at 10,000 bp resolution.

S. aureus

S. aureus consists of one chromosome and two plasmids. The first guiding genome of S. aureus consists of a single chromosome. This produced 3 scaffolds, using 94 of 162 contigs. No chromosomal errors were made and no incorrect orientation or ordering was done. 12 of 91 contig links had wrong gap estimates at 500 bp resolution, and 8 gap estimates were wrong at 10,000 bp resolution.

6.1.2 Choosing an optimal contig end length

In order to assess the effect of extracting contig ends prior to alignment, different contig end *lengths* were tested. *GuideScaff* was run on all datasets with the same parameters as in section 6.1.1.

The plots in fig. 6.1 on page 46 through fig. 6.4 on page 47 shows the effect on *correctness* the resulting scaffolds run had on the different datasets. The correctness measure was used as described in section 4.1.4 on page 31.

The optimal value of contig end length N varied between the datasets, but runs using contigs in full length ($N = 0$) proved to never be an optimal choice in terms of correctness.

For E. coli $N = 1,000$ gave the best correctness score.

For P. suwonensis the best result was with $N = 400$. Figure 6.2 on page 46 shows two scores of 1 at $N = 100$ and $N = 200$, but these were special cases where *no* scaffolds were produced, thus no errors were made.

R. sphaeroides had a maximum correctness score with $N = 1,200$ and S. aureus with $N = 2,000$.

From these results, a common contig end length $N = 1,000$ was chosen for all further experiments when contig ends were to be used.

6.1.3 Using contig ends with a fixed length

Table 6.2 on page 45 shows the results when using contig end extraction with the contig end length fixed at 1,000.

In this run the number of contigs scaffolded increased for all datasets except for P. suwonensis.

N50 values were much smaller for all of the datasets now, compared to the run with no contig ends extracted.

	E. coli	P. suwonensis	R. sphaeroides	S. aureus
Contig end length	0	0	0	0
# Guides	1	1	1	1
Window size	2	2	2	2
Threshold	1	1	1	1
Genome size	4,639,675	3,419,049	4,603,060	2,903,081
# Contigs	481	303	583	162
# Scaffolds	4	3	13	3
N50 scaffolds	2,465,078	3,169,365	2,730,310	2,016,698
# Contigs used	421	97	387	94
Fraction of contigs used	.875	.320	.663	.580
# Pairs of contigs	417	94	374	91
Different chromosomes	0	0	1	0
Different orientations	0	37	3	0
Different order	0	38	2	0
Gap errors > 100	18	85	58	22
Gap errors > 500	15	77	36	12
Gap errors > 1,000	13	73	33	9
Gap errors > 10,000	2	61	18	8

Table 6.1: Initial results of running *GuideScaff* on all 4 datasets. Parameters were set as indicated, and only one guiding genome was used for each dataset. Contigs were used in their entirety, i.e. no contig ends were extracted. The entire report can be seen in table B.2 on page 74.

	E. coli	P. suwonensis	R. sphaeroides	S. aureus
Contig end length	1,000	1,000	1,000	1, 000
# Guides	1	1	1	1
Window size	3	3	3	3
Threshold	1	1	1	1
Genome size	4,639,675	3,419,049	4,603,060	2,903,081
# Contigs	481	303	583	162
# Scaffolds	19	27	64	17
N50 scaffolds	597,757	46,936	84,362	252,200
# Contigs used	433	65	389	98
Fraction of contigs used	.900	.214	.667	.604
# Pairs of contigs	414	38	325	81
Different chromosomes	0	0	2	0
Different orientations	0	6	0	3
Different order	0	5	1	2
Gap errors > 100	16	34	53	31
Gap errors > 500	13	27	43	24
Gap errors > 1,000	12	26	40	21
Gap errors > 10,000	6	21	17	16

Table 6.2: Initial results of running *GuideScaff* on all 4 datasets. Unlike table 6.1, this run used $w = 3$, as it gave better results than $w = 2$ for the same data. The contig end length was fixed at 1,000 bp. The entire report can be seen in table B.3 on page 75.

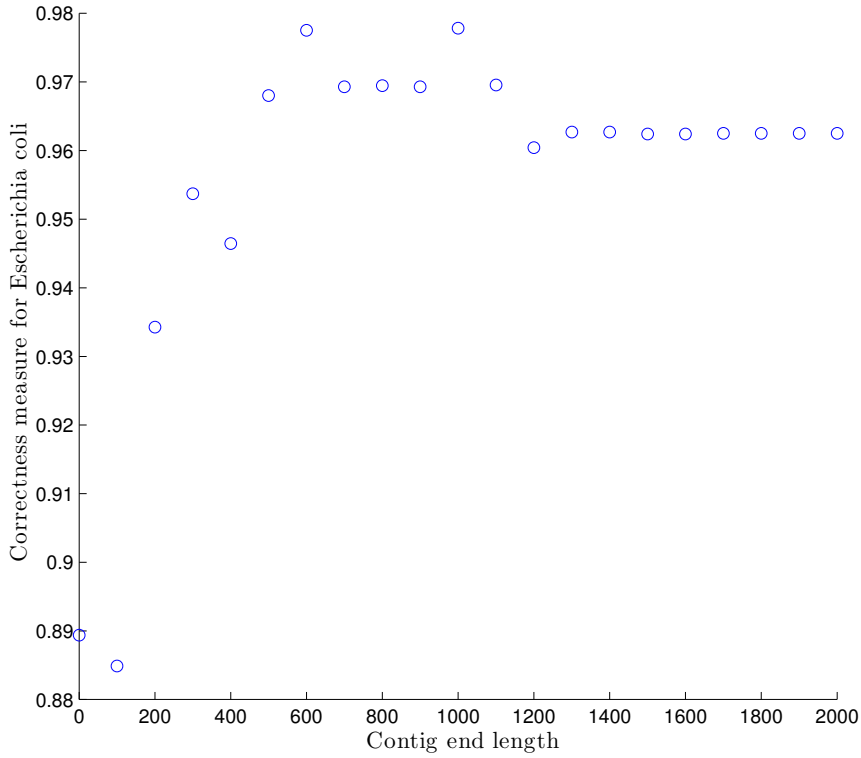

Figure 6.1: Contig end length and correctness for E. coli.

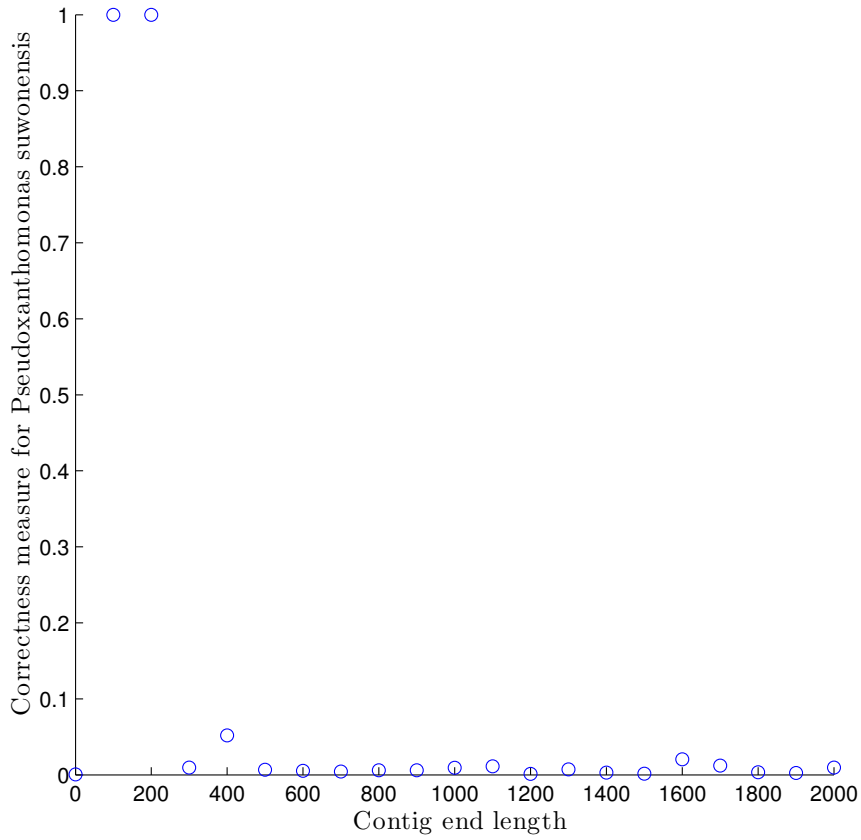

Figure 6.2: Contig end length and correctness for P. suwonensis.

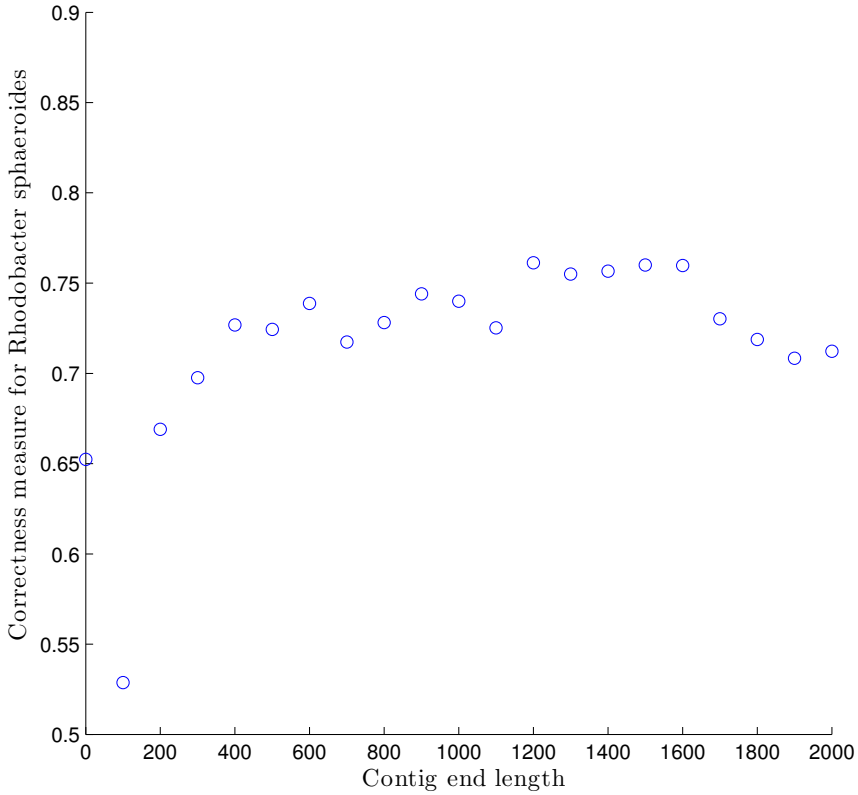

Figure 6.3: Contig end length and correctness for R. sphaeroides.

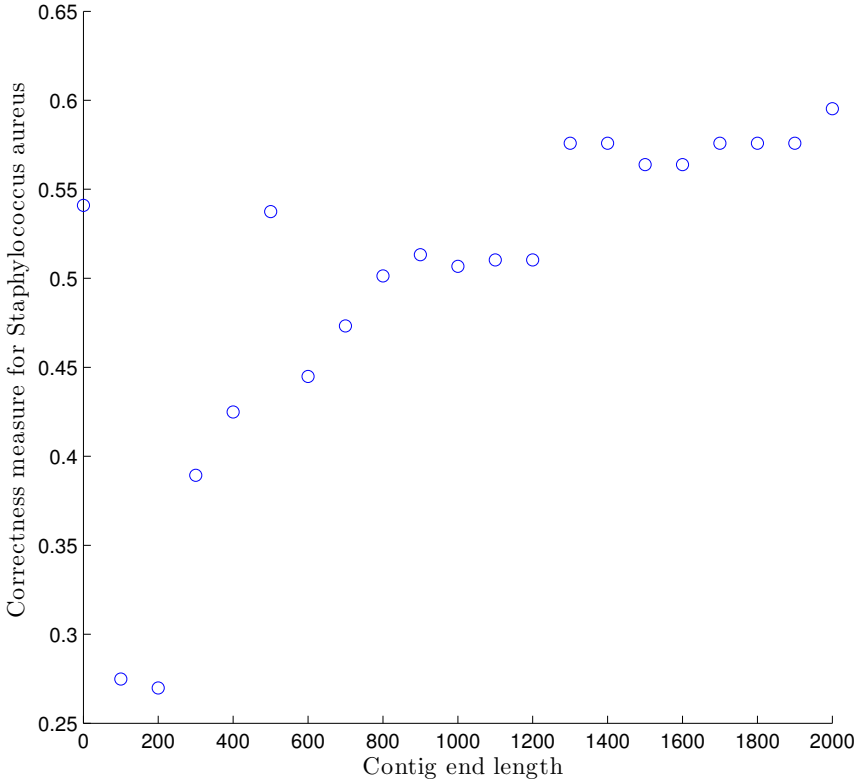

Figure 6.4: Contig end length and correctness for S. aureus.

6.2 Using multiple guiding genomes

This sections shows how *GuideScaff* performed on all the datasets when 9 additional guiding genomes were added, and when using different parameters.

The number of guiding genomes t which must agree affected the output results as shown in the following plots.

First, each dataset was run *without* contig ends extraction. Then it was run with a fixed contig end length $N = 1,000$.

Complete tables of these results are shown in table B.4 on page 76 through table B.7 on page 79.

6.2.1 Without contig ends extraction

When the threshold value t were increased, fewer contigs were scaffolded for each dataset, as shown in fig. 6.5.

When less contigs were scaffolded, the N50 measure also decreased, as shown in fig. 6.6. Here, E. coli got the highest N50 values.

Figure 6.7 on page 50 shows the number of errors made by linking contigs together when they mapped to different chromosomes in the target genome. 25 such errors were made for R. sphaeroides at $t = 1$. This genome had the largest amount of chromosomes and plasmids of the targets, explaining why more errors were done with this bacterium than with the others. When t was increased to 2, only 2 errors remained for R. sphaeroides.

Figure 6.8 on page 50 shows the number of errors made in terms of incorrect relative ordering of the contigs. For R. sphaeroides, this started out with over 90 errors made at $t = 1$, and dropped rapidly to 20 and 0 with $t = 2$ and $t = 3$.

A similar tendency can be seen in fig. 6.9 on page 51, where the number of contigs in incorrect relative orientations are shown. Again, a large number of errors were made with $t = 1$, but for all datasets the number of errors decreased when t was increased.

The errors of gap estimates exceeding $\Delta = 500$ and $\Delta = 10,000$ are shown in fig. 6.10 on page 51 and fig. 6.11 on page 52. These errors were also *mostly* decreasing with an increasing threshold value, except for the scaffolds produced for E. coli with 500 bp resolution. At this resolution, the scaffolds of E. coli had least gap errors when $t = 3$. This was one of few exceptions to error decrease when the threshold value was increased.

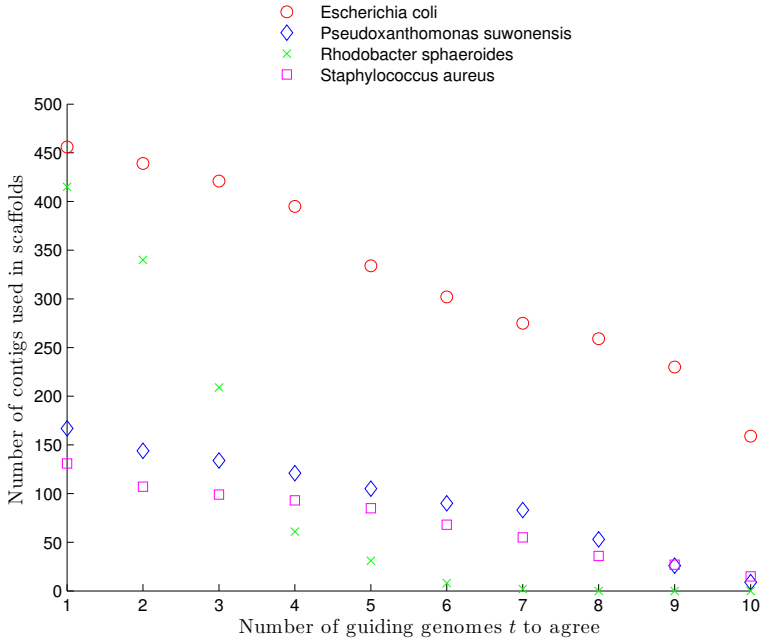

Figure 6.5: Threshold and number of contigs used in scaffolds produced. As more genomes had to agree on contig links, the number of contigs used in the scaffolds decreased.

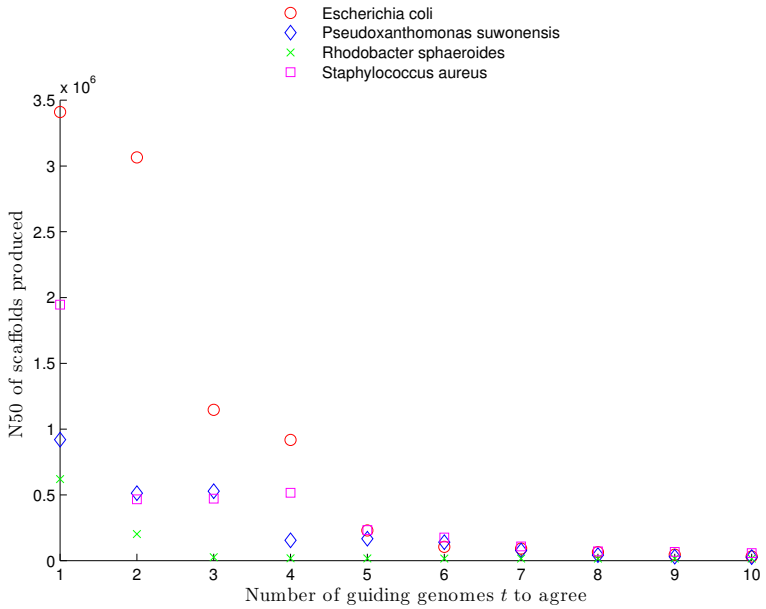

Figure 6.6: Threshold and N50 of scaffolds produced. With an increasing threshold value, the resulting N50 metrics of the produced scaffolds mostly decreased.

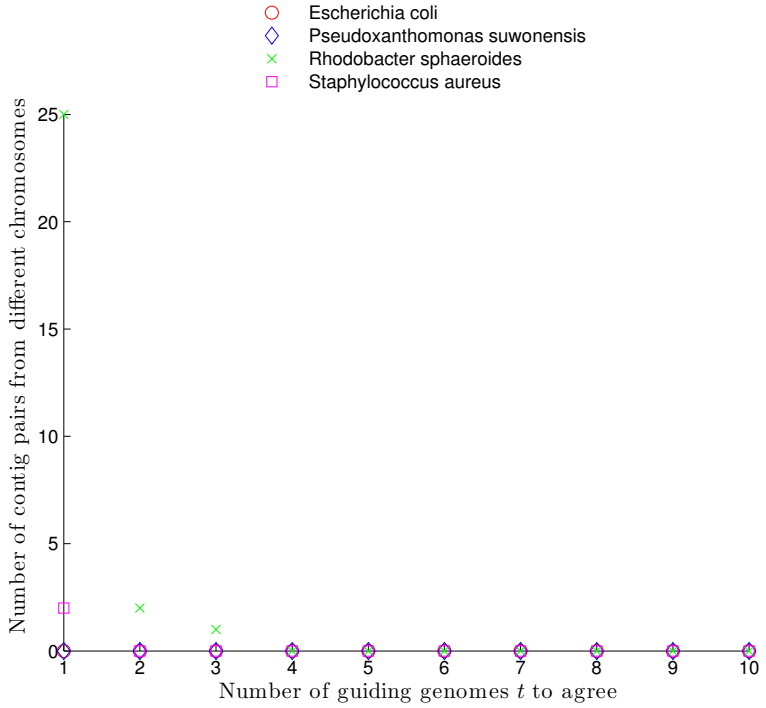

Figure 6.7: Threshold and number of contig pairs in different chromosomes. The number of contig links mapping to different chromosomes in the target genome decreased as the threshold was increased. From $t = 5$, no such errors were made.

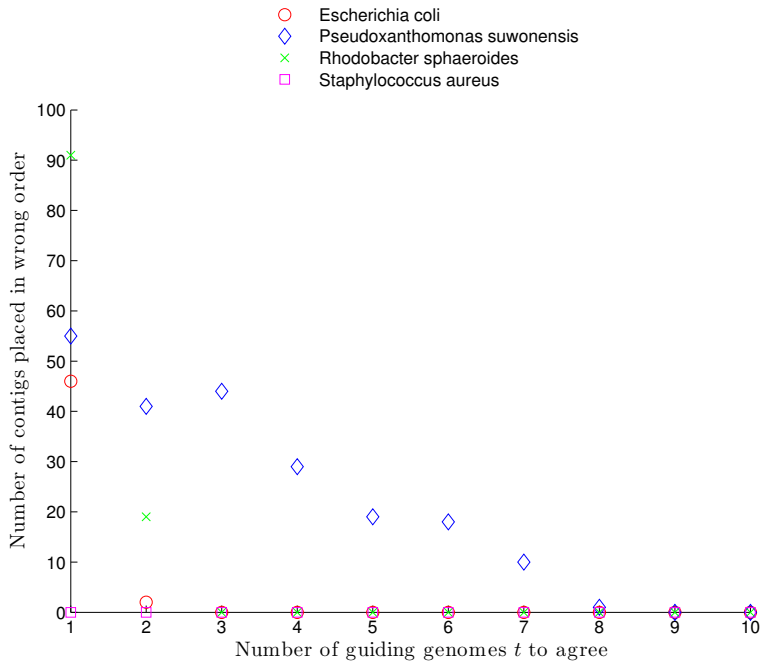

Figure 6.8: Threshold and number of contigs in incorrect order. This error type mostly decreased when t was increased.

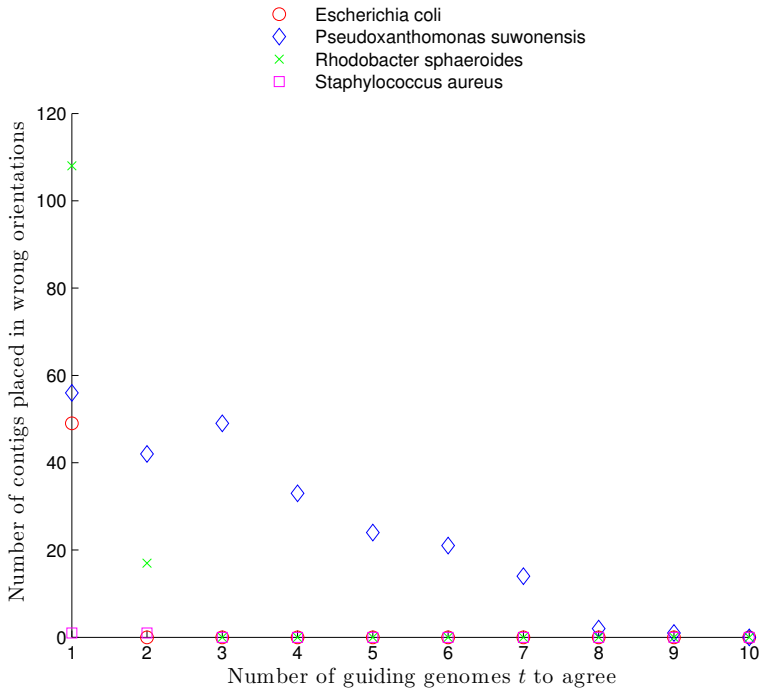

Figure 6.9: Threshold and number of contigs in incorrect orientation. This error type also decreased as t was increased.

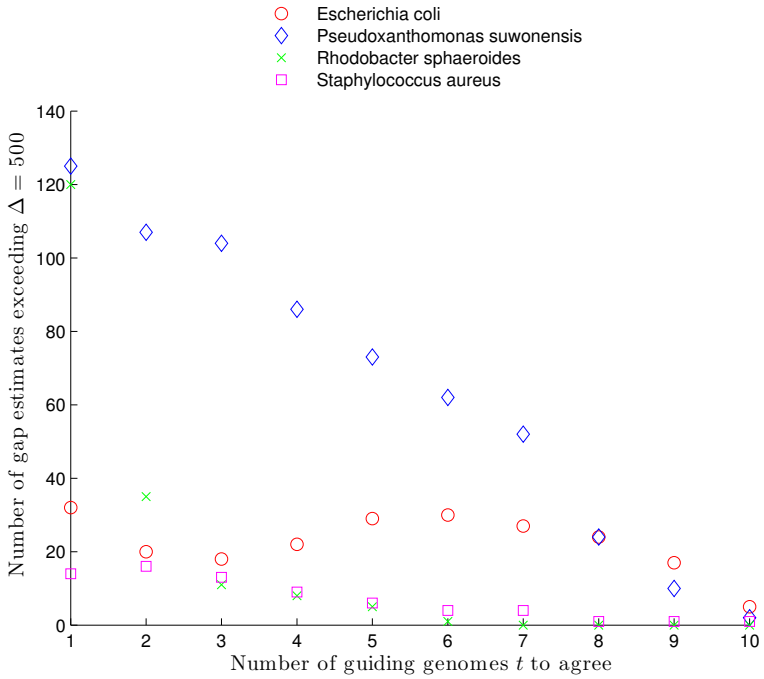

Figure 6.10: Threshold and number of gap estimate errors > 500 bp. This error type mostly decreased as t was increased.

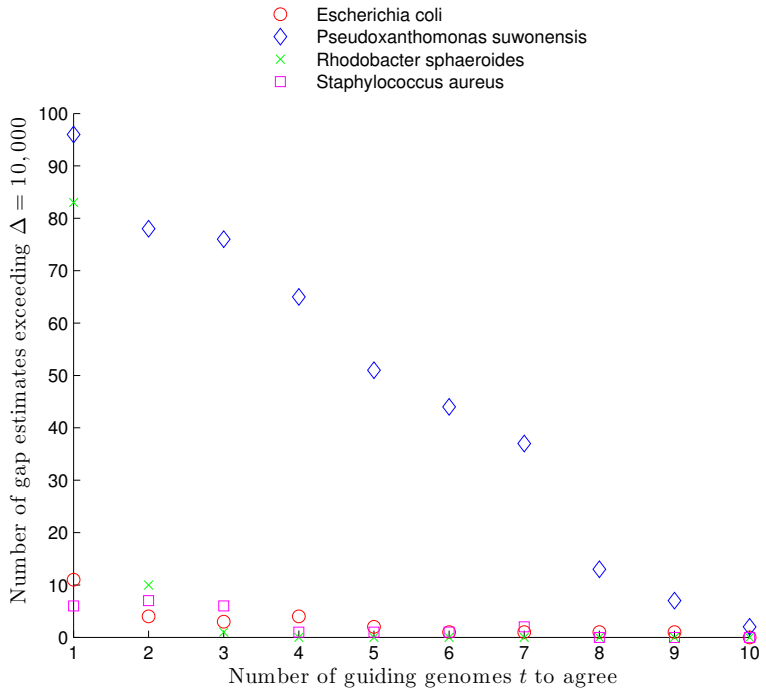

Figure 6.11: Threshold and number of gap estimate errors > 10,000 bp. This error type mostly decreased as t was increased.

6.2.2 With contig ends extraction

The following plots shows the same measures as the previous 7 plots, except that contig end extraction was used with $N = 1,000$.

Complete tables of these results are shown in table B.8 on page 80 through table B.11 on page 83.

Figure 6.12 shows the number of contigs used in the resulting scaffolds, when contig end extraction were used with $N = 1,000$. It shows a similar tendency as fig. 6.5 on page 49 did, with a general decrease of contigs used when the threshold value was increased.

The same tendency can be seen for the N50 measure of the scaffolds produced, shown in fig. 6.13.

The number of chromosomal errors made are shown in fig. 6.14. This shows that fewer errors of this type were made when contig ends were used, than when the entire contigs were used.

Figure 6.15 on page 55 shows the number contigs placed in incorrect order in the resulting scaffolds. This error type rapidly decreased from $t = 1$ to $t = 2$.

The same tendency can be shown in fig. 6.16 on page 55. Here, the number of contigs incorrectly oriented fell rapidly with an increasing t.

The errors made in gap estimates at the two plotted resolutions, 500 bp and 10,000 bp, also conformed to this tendency. An exception to this was the resulting scaffolds for E. coli, for which gap errors decreased, but not in the same strict fashion as for the other datasets.

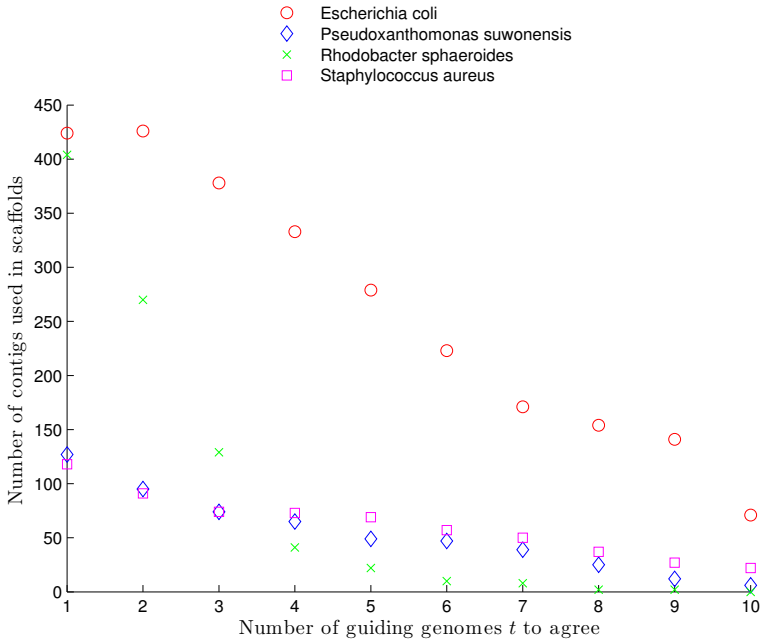

Figure 6.12: Increasing threshold and number of contigs in scaffolds. Contig end length $N = 1,000$. The number of contigs used decreased as the threshold value was increased.

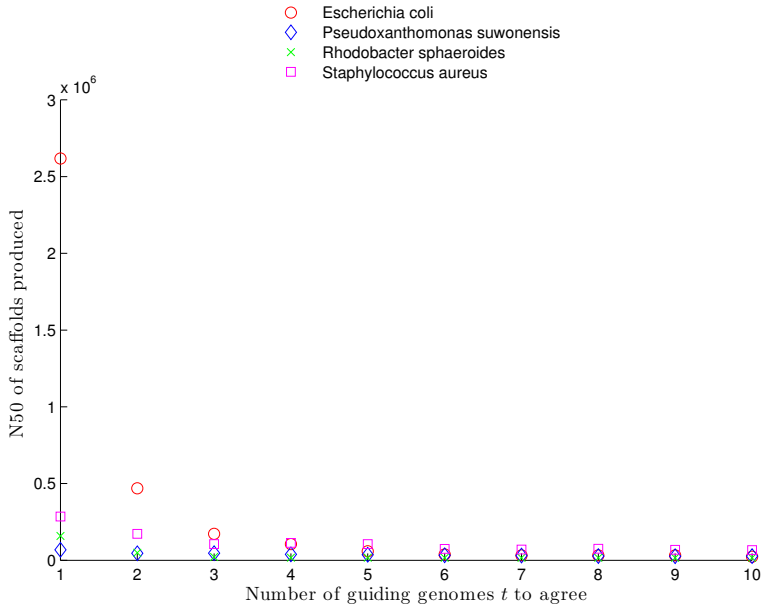

Figure 6.13: Increasing threshold and N50 of produced scaffolds. Contig end length $N = 1{,}000$. This metric decreased as the threshold was increased.

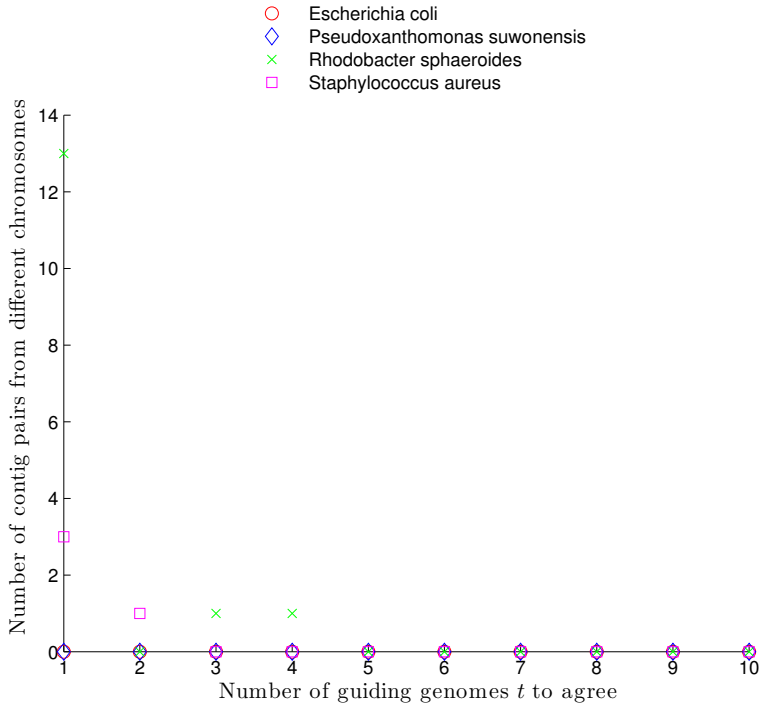

Figure 6.14: Increasing threshold and number of contig pairs mapping to different chromosomes. Contig end length $N = 1{,}000$. This error type decreased as the threshold was increased. From $t = 5$ no such errors were made.

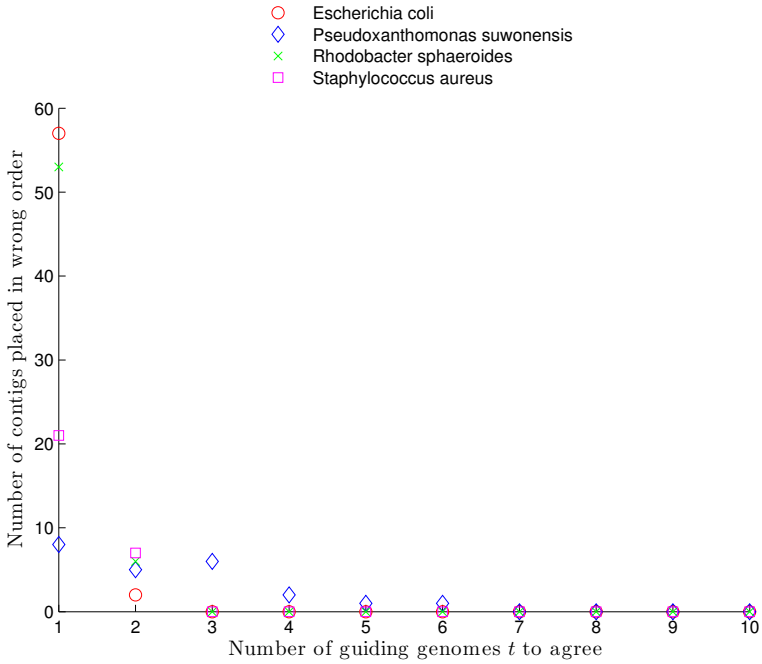

Figure 6.15: Increasing threshold and number of contigs used in wrong relative order. Contig end length $N = 1,000$. This error type decreased as the threshold was increased. From $t = 7$ no such errors were made.

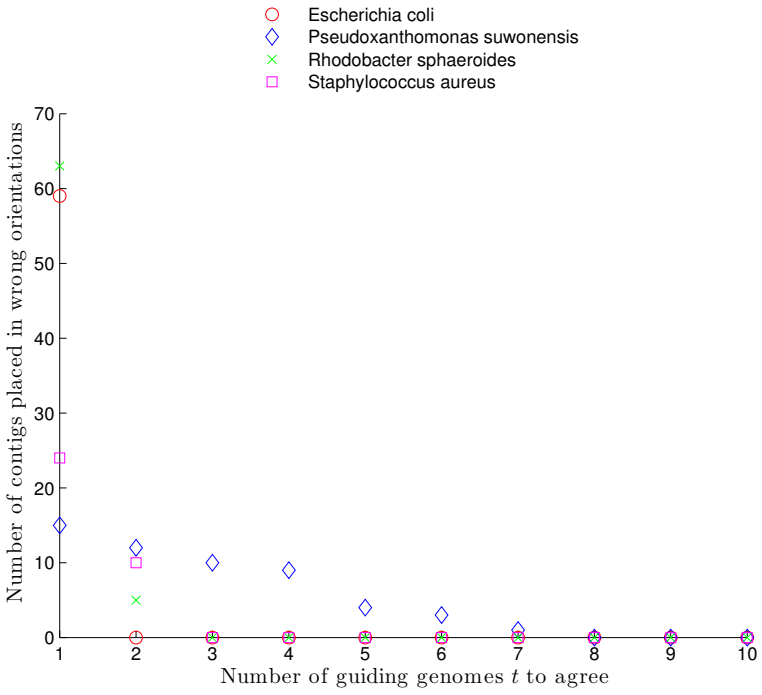

Figure 6.16: Increasing threshold and number of contigs used in wrong relative orientation. Contig end length $N = 1,000$. From $t = 8$ no such errors were made.

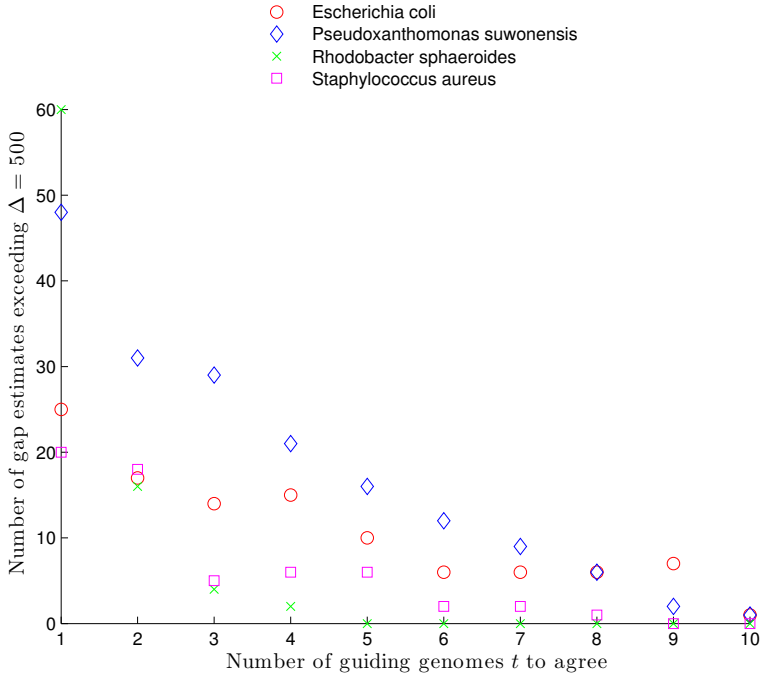

Figure 6.17: Threshold and number of gap estimate errors > 500 bp. This type of error mostly decreased with an increasing t, but was never completely removed from all the datasets.

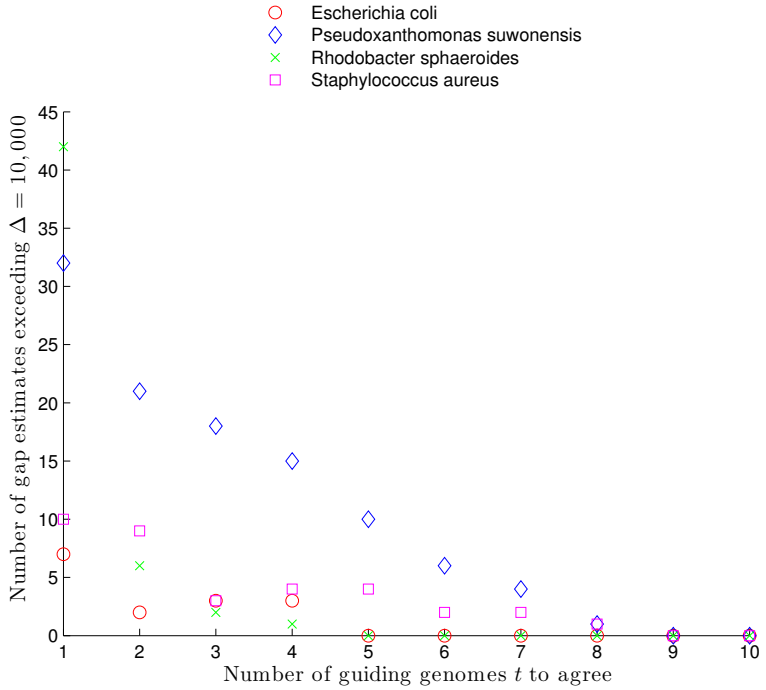

Figure 6.18: Threshold and number of gap estimate errors > 10,000 bp. This type of error also mostly decreased with an increasing t, and no such errors were made from $t = 9$.

Chapter 7

Discussion

7.1 Interpreting the results

7.1.1 Initial runs on all datasets

When *GuideScaff* was run on all datasets using *one* guiding genome and no contig ends extraction, few errors were made for E. coli, R. sphaeroides and S. aureus, but many errors were made for P. suwonensis.

The fraction of contigs used in the resulting scaffolds spanned from 32% to 87.5%.

The high error rate when scaffolding P. suwonensis and a chromosomal error of the scaffolds for R. sphaeroides makes it interesting to test this method using several guiding genomes. This makes it possible to increase the number of contigs scaffolded. This also enables the threshold of agreeing genomes to be adjusted, possibly lowering the error rates.

Comparison with Velvet scaffolds with mate-pairs

The N50 measure of the scaffolds produced by *GuideScaff* spanned from about 2 million bp to about 4.6 million bp. The corresponding N50 values for the scaffolds produced by Velvet (table 5.4 on page 35) using mate-pairs spanned from about 57,000 bp to about 762,000 bp.

7.1.2 Extracting contig ends

The main idea of contig extraction was to reduce the ambiguity of long contigs when aligned to the guiding genomes.

As figures fig. 6.1 on page 46 through fig. 6.4 on page 47 shows, the correctness measure had maxima at different contig end lengths, but never at $N = 0$. This means that in terms of correctness, contig end extraction proved to be beneficial in these cases, but *one* fixed length were not optimal for all the genomes.

A contig end length of 1,000 bp was chosen in the further experiments, as this was approximately the average value of the 4 correctness maxima from these plots.

7.1.3 Using multiple guiding genomes

Without contig end extraction

When 9 additional guiding genomes were added, the number of scaffolded contigs for E. coli increased from 421 to 456. For P. suwonensis it increased from 97 to 127, for R. sphaeroides from 387 to 404 and for S. aureus from 94 to 118.

The increase in number of guiding genomes used also increased the amount of errors made. However, a trend proved to be common for all error types: As more of the genomes had to agree on contig links, both chromosomal errors, positioning and ordering errors decreased. Gap errors at all the resolutions were also decreased when the threshold was increased.

With contig end extraction

Using contig end extraction it proved to be possible to lower the error rates of the scaffolds produced when the guiding genomes used were highly dissimilar to the target genome. Fewer contigs were scaffolded than without contig end extraction, but the relative error rates decreased.

7.2 Analysis of the proposed method

7.2.1 Performance

GuideScaff works for scaffolding contigs based on related genomes. It can handle an arbitrary number of related genomes, and use contig alignments on them, either on a nucleotide level or on a protein level.

The precision of the resulting scaffolds can be increased with an increasing threshold value t, forcing at least t of the guiding genomes used to agree on contig links.

The most computational intensive parts of the proposed method lies within the alignments which are done by tools from MUMmer. If window size w is fixed, the procedures of processing tiling files and building contig links grows linearly in time.

7.2.2 Potential usage

When the guiding genomes are closely related to the target genome, *GuideScaff* is able to produce scaffolds using most of the contigs at hand. By forcing at least two guiding genomes to agree on contig links, error rates are dramatically lowered.

When using more distant related genomes, error rates are higher, but decreasing when parameters are set more strict.

The method could be used as it is to create candidate scaffolds in *de novo* sequencing projects where at least one related guiding genome is available.

GuideScaff could be used as a supplement to other strategies for scaffolding in sequencing projects.

Unlike the use of mate-pairs, a scaffolding procedure guided by related genomes as shown in this thesis is completely free of cost, and it runs fast.

7.2.3 Weaknesses

The main weakness of *GuideScaff* is its greediness. When contigs are linked, only local optima are considered. This can result in incorrect gap estimates as well as incorrect contig links. With a non-greedy algorithm, global measures such as the total genome length could be used to make optimal choices all through the scaffolding process.

Mate-pairs are often used to produce scaffolds, but they are not utilized in *GuideScaff*. If mate-pairs are available, the method could most likely be improved by considering the information these mate-pairs provide in addition to the alignments on the guiding genomes.

The contig end length should *not* be static, as the experiments show that different contig end lengths works better for individual datasets. This could be solved by finding multiple high-quality alignments for each contig and then automatically select the first left-most and right-most alignments as contig ends.

GuideScaff weighs all guiding genomes equally. This could cause gap estimates to be incorrect if some of the guiding genomes are highly dissimilar to the target genome. This is partially fixed by using a small window size when traversing the tilings list, and by using the *median* of the distances reported. However, if a larger window size is used to increase the scaffold sizes, the median of a larger number of distances may not suffice as a gap estimate.

The threshold value of 90% average identity to determine whether to use nucleotide- or protein alignment is set based on a series of tests, and is never proven to be optimal for any of the datasets. This threshold could be set after more thorough experiments.

When protein alignment is chosen, this is done using the default alignment matrix for `promer`, which is BLOSUM62. Other alignment matrices could be determined to be used in order to provide better alignments for each guiding genome.

7.3 Further work

As the use of guiding genomes in scaffolding has proven to be worth looking into, a number of improvements could be made to the proposed method. This includes

- Detection and removal of outliers in reported distances between contigs when using large window sizes
- Dynamically find optimal contig ends, without a fixed length

- Automatically choose appropriate ways of protein alignment involving different scoring matrices
- Utilizing mate-pair as supplementary information when available
- Using a weighting scheme on distance estimates based on sequence similarities
- Implementing a non-greedy algorithm with a global optimization scheme

7.4 Conclusion

The proposed and tested methods of *GuideScaff* works as a proof of concept, showing that related genomes can be useful in scaffolding.

If the available guiding genomes are closely related to the target, one such genome can suffice. In other cases, many guiding genomes can be preferable to a few, both in order to increase the number of contigs scaffolded and to minimize the number of errors.

When dealing with distantly related sequences, extracting contig ends prior to alignment increases the scaffold correctness. The experiments show that contig end extraction produces scaffolds with lower error-rates than if the entire contigs are aligned.

When using multiple guiding genomes, demanding *at least two* of the guiding genomes to agree, significantly increases the scaffold correctness.

With the rapid growth of completely sequenced genomes available, the use of related genomes will be increasingly helpful in the process of scaffolding. Such methods provide an inexpensive alternative to additional lab-work.

Bibliography

Journal papers

[1] S. Assefa, T. M. Keane, T. D. Otto, C. Newbold, and M. Berriman. "ABACAS: algorithm-based automatic contiguation of assembled sequences." eng. In: *Bioinformatics (Oxford, England)* 25.15 (Aug. 2009). PMID: 19497936, pp. 1968–1969 (cit. on p. 13).

[3] M. Boetzer, C. V. Henkel, H. J. Jansen, D. Butler, and W. Pirovano. "Scaffolding pre-assembled contigs using SSPACE." eng. In: *Bioinformatics (Oxford, England)* 27.4 (Feb. 2011). PMID: 21149342, pp. 578–579 (cit. on p. 13).

[4] K. R. Bradnam, J. N. Fass, A. Alexandrov, P. Baranay, M. Bechner, İ. Birol, S. Boisvert10, J. A. Chapman, G. Chapuis, R. Chikhi, H. Chitsaz, W.-C. Chou, J. Corbeil, C. Del Fabbro, T. R. Docking, R. Durbin, D. Earl, S. Emrich, P. Fedotov, N. A. Fonseca, G. Ganapathy, R. A. Gibbs, S. Gnerre, É. Godzaridis, S. Goldstein, M. Haimel, G. Hall, D. Haussler, J. B. Hiatt, I. Y. Ho, J. Howard, M. Hunt, S. D. Jackman, D. B. Jaffe, E. Jarvis, H. Jiang, S. Kazakov, P. J. Kersey, J. O. Kitzman, J. R. Knight, S. Koren, T.-W. Lam, D. Lavenier, F. Laviolette, Y. Li, Z. Li, B. Liu, Y. Liu, R. Luo, I. MacCallum, M. D. MacManes, N. Maillet, S. Melnikov, B. M. Vieira, D. Naquin, Z. Ning, T. D. Otto, B. Paten, O. S. Paulo, A. M. Phillippy, F. Pina-Martins, M. Place, D. Przybylski, X. Qin, C. Qu, F. J. Ribeiro, S. Richards, D. S. Rokhsar, J. G. Ruby, S. Scalabrin, M. C. Schatz, D. C. Schwartz, A. Sergushichev, T. Sharpe, T. I. Shaw, J. Shendure, Y. Shi, J. T. Simpson, H. Song, F. Tsarev, F. Vezzi, R. Vicedomini, J. Wang, K. C. Worley, S. Yin, S.-M. Yiu, J. Yuan, G. Zhang, H. Zhang, S. Zhou, and I. F. Korf1. "Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species." In: *arXiv:1301.5406* (Jan. 2013) (cit. on p. 26).

[6] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice. "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants." eng. In: *Nucleic acids research* 38.6 (Apr. 2010). PMID: 20015970, pp. 1767–1771 (cit. on p. 15).

[7] P. E. C. Compeau, P. A. Pevzner, and G. Tesler. "How to apply de Bruijn graphs to genome assembly." eng. In: *Nature biotechnology* 29.11 (Nov. 2011). PMID: 22068540, pp. 987–991 (cit. on p. 12).

[8] A. E. Darling, B. Mau, and N. T. Perna. “progressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement.” In: *PLoS ONE* 5.6 (June 2010), e11147 (cit. on p. 36).

[9] A. E. Darling, B. Mau, and N. T. Perna. “progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement.” eng. In: *PloS one* 5.6 (2010). PMID: 20593022, e11147 (cit. on p. 67).

[10] J. C. Dohm, C. Lottaz, T. Borodina, and H. Himmelbauer. “Substantial biases in ultra-short read data sets from high-throughput DNA sequencing.” eng. In: *Nucleic acids research* 36.16 (Sept. 2008). PMID: 18660515, e105 (cit. on p. 10).

[11] S. R. Eddy. “What is dynamic programming?” eng. In: *Nature biotechnology* 22.7 (July 2004). PMID: 15229554, pp. 909–910 (cit. on p. 13).

[12] S. Gao, W.-K. Sung, and N. Nagarajan. “Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences.” eng. In: *Journal of computational biology: a journal of computational molecular cell biology* 18.11 (Nov. 2011). PMID: 21929371, pp. 1681–1691 (cit. on p. 13).

[13] M. B. Gerstein, C. Bruce, J. S. Rozowsky, D. Zheng, J. Du, J. O. Korbel, O. Emanuelsson, Z. D. Zhang, S. Weissman, and M. Snyder. “What is a gene, post-ENCODE? History and updated definition.” eng. In: *Genome research* 17.6 (June 2007). PMID: 17567988, pp. 669–681 (cit. on p. 5).

[14] A. A. Gritsenko, J. F. Nijkamp, M. J. T. Reinders, and D. de Ridder. “GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies.” eng. In: *Bioinformatics (Oxford, England)* 28.11 (June 2012). PMID: 22492642, pp. 1429–1437 (cit. on pp. 13, 29, 33–35).

[17] D. R. Kelley, M. C. Schatz, and S. L. Salzberg. “Quake: quality-aware detection and correction of sequencing errors.” eng. In: *Genome biology* 11.11 (2010). PMID: 21114842, R116 (cit. on pp. 10, 21).

[18] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S. L. Salzberg. “Versatile and open software for comparing large genomes.” eng. In: *Genome biology* 5.2 (2004). PMID: 14759262, R12 (cit. on p. 21).

[19] E. R. Mardis. “A decade/’s perspective on DNA sequencing technology.” en. In: *Nature* 470.7333 (Feb. 2011), pp. 198–203 (cit. on p. 9).

[20] E. Merino, P. Balbás, J. L. Puente, and F. Bolívar. “Antisense overlapping open reading frames in genes from bacteria to humans.” eng. In: *Nucleic acids research* 22.10 (May 1994). PMID: 8208617, pp. 1903–1908 (cit. on p. 20).

[22] E. W. Myers, G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H. H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G. M. Rubin, M. D. Adams, and J. C. Venter. "A whole-genome assembly of Drosophila." eng. In: *Science (New York, N.Y.)* 287.5461 (Mar. 2000). PMID: 10731133, pp. 2196–2204 (cit. on p. 12).

[23] H. Pearson. "Genetics: what is a gene?" eng. In: *Nature* 441.7092 (May 2006). PMID: 16724031, pp. 398–401 (cit. on p. 5).

[24] M. Pop. "Genome assembly reborn: recent computational challenges." eng. In: *Briefings in bioinformatics* 10.4 (July 2009). PMID: 19482960, pp. 354–366 (cit. on pp. 7, 9, 11, 12).

[25] M. Pop, D. S. Kosack, and S. L. Salzberg. "Hierarchical scaffolding with Bambus." eng. In: *Genome research* 14.1 (Jan. 2004). PMID: 14707177, pp. 149–159 (cit. on p. 13).

[26] L. Salmela, V. Mäkinen, N. Välimäki, J. Ylinen, and E. Ukkonen. "Fast scaffolding with small independent mixed integer programs." eng. In: *Bioinformatics (Oxford, England)* 27.23 (Dec. 2011). PMID: 21998153, pp. 3259–3265 (cit. on p. 13).

[27] S. L. Salzberg, A. M. Phillippy, A. Zimin, D. Puiu, T. Magoc, S. Koren, T. J. Treangen, M. C. Schatz, A. L. Delcher, M. Roberts, G. Marçais, M. Pop, and J. A. Yorke. "GAGE: A critical evaluation of genome assemblies and assembly algorithms." eng. In: *Genome research* 22.3 (Mar. 2012). PMID: 22147368, pp. 557–567 (cit. on pp. 21, 33–35).

[28] F. Sanger, S. Nicklen, and A. R. Coulson. "DNA sequencing with chain-terminating inhibitors. 1977." eng. In: *Biotechnology (Reading, Mass.)* 24 (1992). PMID: 1422003, pp. 104–108 (cit. on p. 9).

[29] O. Tange. "GNU Parallel - The Command-Line Power Tool." In: *;login: The USENIX Magazine* 36.1 (Feb. 2011), pp. 42–47 (cit. on p. 28).

[31] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, R. Edgar, S. Federhen, M. Feolo, L. Y. Geer, W. Helmberg, Y. Kapustin, O. Khovayko, D. Landsman, D. J. Lipman, T. L. Madden, D. R. Maglott, V. Miller, J. Ostell, K. D. Pruitt, G. D. Schuler, M. Shumway, E. Sequeira, S. T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, R. L. Tatusov, T. A. Tatusova, L. Wagner, and E. Yaschenko. "Database resources of the National Center for Biotechnology Information." eng. In: *Nucleic acids research* 36.Database issue (Jan. 2008). PMID: 18045790, pp. D13–21 (cit. on pp. 1, 13, 34).

[33] D. R. Zerbino and E. Birney. “Velvet: algorithms for de novo short read assembly using de Bruijn graphs.” eng. In: *Genome research* 18.5 (May 2008). PMID: 18349386, pp. 821–829 (cit. on pp. 12, 21, 34, 35).

[34] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. “A greedy algorithm for aligning DNA sequences.” eng. In: *Journal of computational biology: a journal of computational molecular cell biology* 7.1-2 (Apr. 2000). PMID: 10890397, pp. 203–214 (cit. on p. 21).

Books

[2] K. Berman. *Algorithms: sequential, parallel, and distributed*. Boston, Mass: Thomson/Course Technology, 2005. ISBN: 978-0-534-42057-4 (cit. on pp. 12, 21).

[5] R. Brooker. *Biology*. New York London: McGraw-Hill Higher Education McGraw-Hill distributor, 2007. ISBN: 978-0-07-110200-1 (cit. on pp. 2–5, 17).

[15] P. Grossman. *Discrete Mathematics for Computing*. Basingstoke: Palgrave Macmillan, 2002. ISBN: 0333981111 (cit. on p. 12).

[32] J. Xiong. *Essential Bioinformatics*. Cambridge University Press, 2006. ISBN: 0521600820 (cit. on pp. 6, 13, 14).

Online references

[16] D. Harper. *Online Etymology Dictionary*. Apr. 24, 2013. URL: http://www.etymonline.com/ (cit. on p. 7).

[21] *Merriam-Webster's Dictionary*. Apr. 24, 2013. URL: http://www.merriam-webster.com/dictionary/ (cit. on pp. 11, 20).

[30] T. Tao. *Single Letter Codes*. Apr. 24, 2013. URL: http://www.ncbi.nlm.nih.gov/staff/tao/tools/tool_lettercode.html (cit. on p. 15).

Appendix

Appendix A

Mauve genome similarity plots

The following plots were created with `progressiveMauve` from Mauve [9]. The red vertical bars in the plots indicates chromosome/plasmid boundaries.

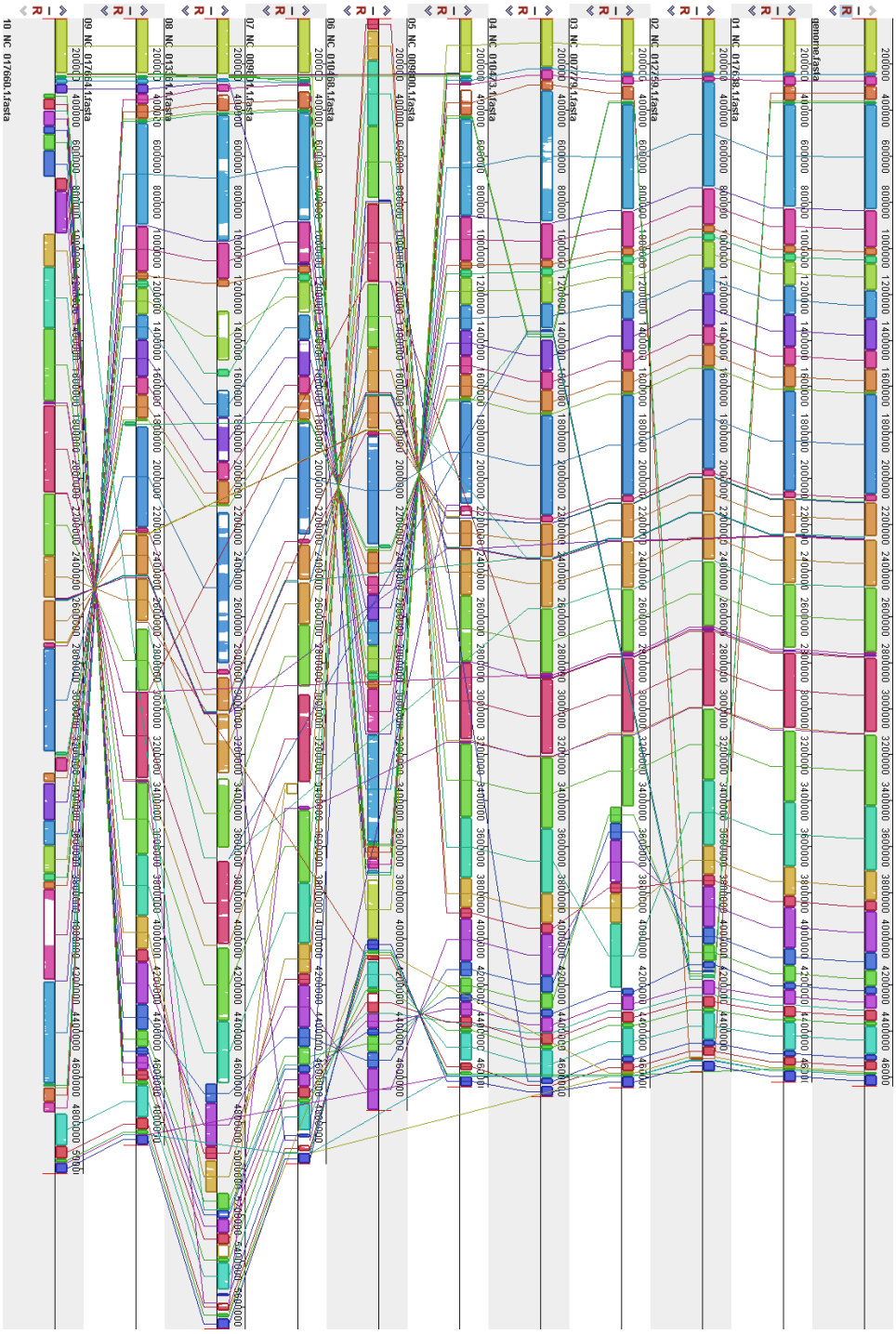

Figure A.1: Mauve alignment of E. coli and the guiding genomes used

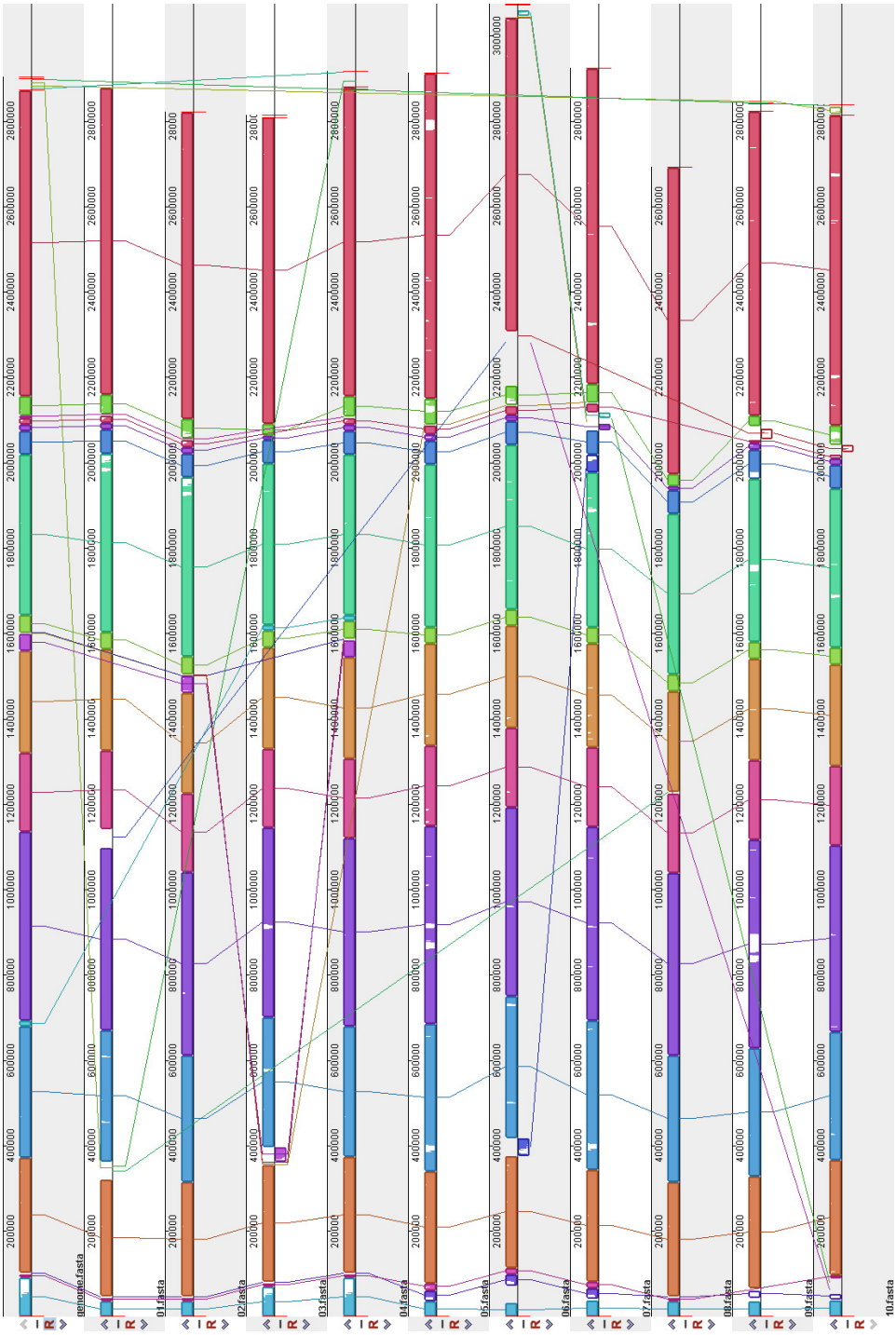

Figure A.2: Mauve alignment of S. aureus and the guiding genomes used

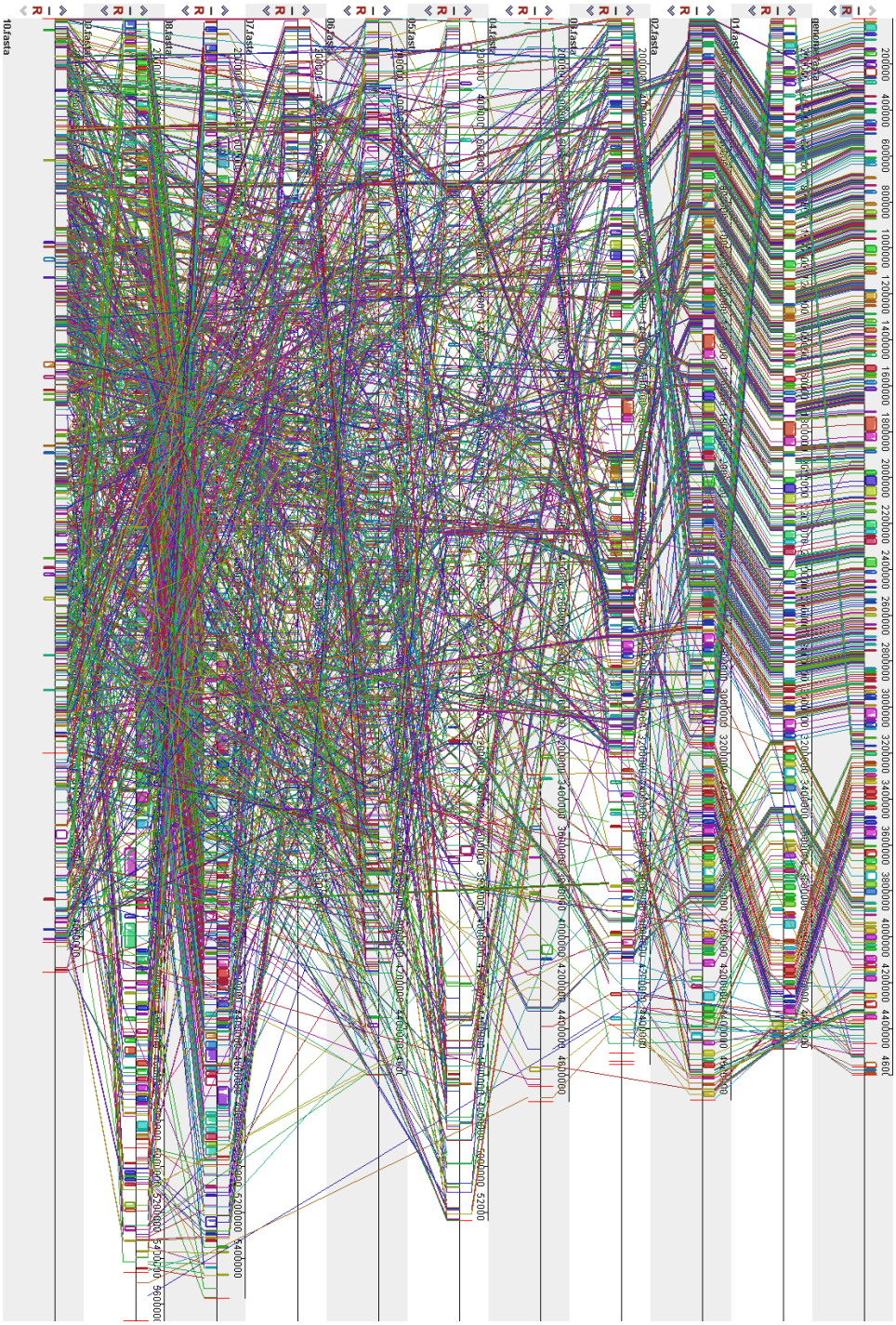

Figure A.3: Mauve alignment of R. sphaeroides and the guiding genomes used

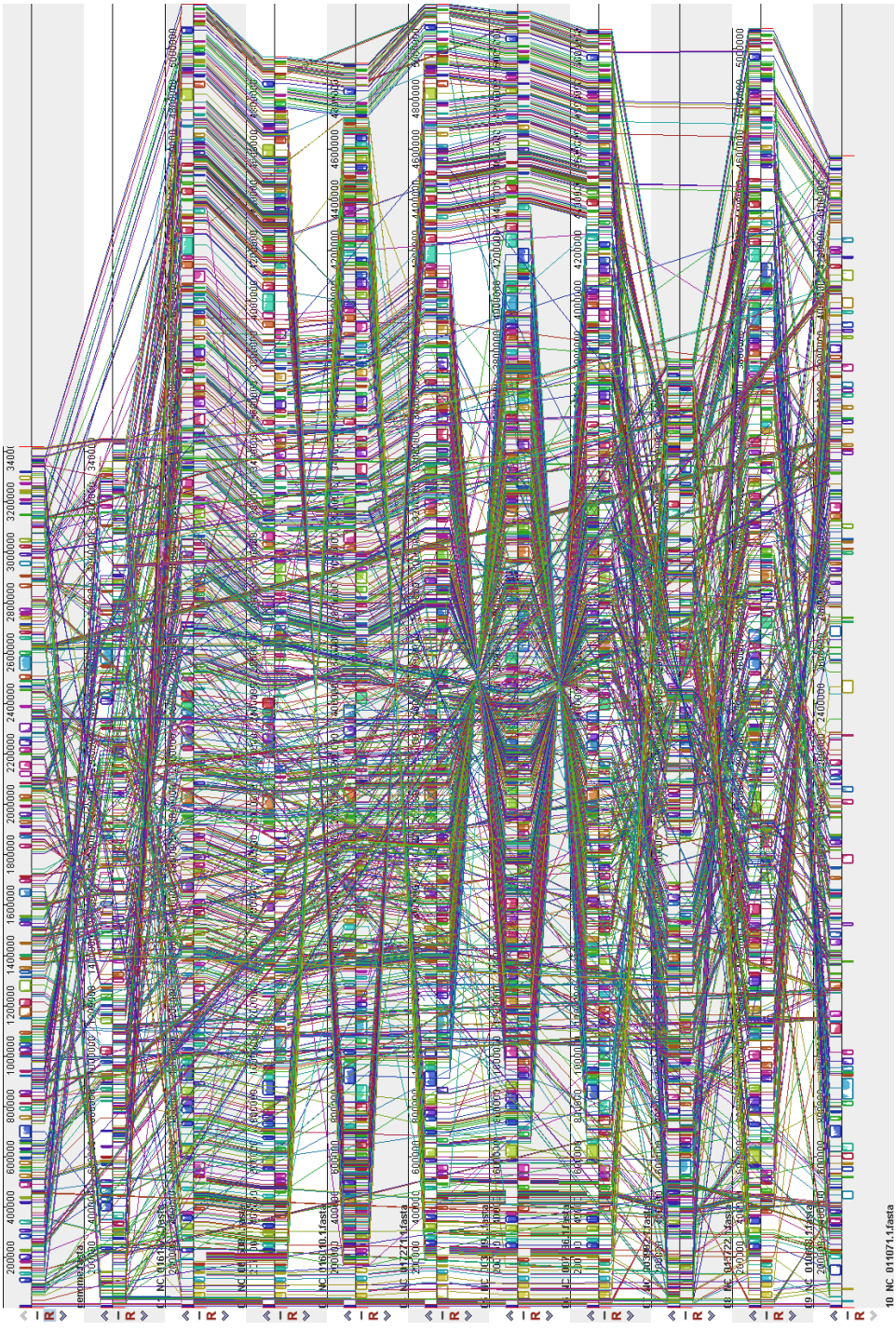

Figure A.4: Mauve alignment of P. suwonensis and the guiding genomes used

Appendix B

Result tables

Field name	Explanation
NCUT	Contig end length
NGUIDES	Number of guiding genomes used
WINDOWSIZE	Size of sliding window used when creating distance matrix
THRESHOLD	Minimum number of agreeing guiding genomes
TARGET GENOME	Name of genome to be scaffolded
GENOME SIZE	Number of bases present in the target genome
CONTIG SET	Filename of contigs used
NCONTIGS	Number of contigs in the contig set
MIN CONTIG LENGTH	Size of smallest contig
MAX CONTIG LENGTH	Size of largest contig
NSCAFFOLDS	Number of scaffolds created
MIN SCAFFOLD LENGTH	Size of smallest scaffold
MAX SCAFFOLD LENGTH	Size of largest scaffold
COVERAGE	Target genome size divided by total scaffold size
N50 CONTIGS	N50 metric for the contigs
N50 SCAFFOLDS	N50 metric for the scaffolds produced
N CONTIGS USED	Number of contigs used in one of the scafffolds produced
REL CONTIGS USED	Fraction of contigs used in one of the scaffolds
N PAIRS	Number of contig links in the scaffolds
DIFF CHROMOSOMES	Number of contig links which maps to different chromosomes in the target genome
DIFF ORIENTATION	Number of contigs in incorrect orientation within the scaffolds
DIFF ORDER	Number of contigs in incorrect order within the scaffolds
GAP ERROR 100	Number of gap estimate errors exceeding 100 bp
GAP ERROR 500	Number of gap estimate errors exceeding 500 bp
GAP ERROR 1000	Number of gap estimate errors exceeding 1,000 bp
GAP ERROR 10000	Number of gap estimate errors exceeding 10,000 bp
REL DIFF CHROMOSOMES	Relative to number of pairs
REL DIFF ORIENTATION	Relative to number of contigs used
REL DIFF ORDER	Relative to number of contigs used
REL GAP ERROR 100	Relative to number of pairs
REL GAP ERROR 500	Relative to number of pairs
REL GAP ERROR 1000	Relative to number of pairs
REL GAP ERROR 10000	Relative to number of pairs

Table B.1: Explanation of the field names of the following result tables.

NCUT	0	0	0	0
NGUIDES	1	1	1	1
WINDOWSIZE	2	2	2	2
THRESHOLD	1	1	1	1
TARGET GENOME	Escherichia coli	Pseudoxanthomonas suwonensis	Rhodobacter sphaeroides	Staphylococcus aureus
GENOME SIZE	4639675	3419049	4603060	2903081
NCONTIGS	481	303	583	162
MIN CONTIG LENGTH	152	151	205	203
MAX CONTIG LENGTH	73062	90572	60714	169214
NSCAFFOLDS	4	3	13	3
MIN SCAFFOLD LENGTH	373781	54746	507	247401
MAX SCAFFOLD LENGTH	2465078	3169365	2730310	2016698
COVERAGE	1.03	1.48	1.16	1.19
N50 CONTIGS	19872	26043	15801	52792
N50 SCAFFOLDS	2465078	3169365	2730310	2016698
N CONTIGS USED	421	97	387	94
REL CONTIGS USED	.875	.320	.663	.580
N PAIRS	417	94	374	91
DIFF CHROMOSOMES	0	0	1	0
DIFF ORIENTATION	0	37	3	0
DIFF ORDER	0	38	2	0
GAP ERROR 100	18	85	58	22
GAP ERROR 500	15	77	36	12
GAP ERROR 1000	13	73	33	9
GAP ERROR 10000	2	61	18	8
REL DIFF CHROMOSOMES	0.000000	0.000000	0.002584	0.000000
REL DIFF ORIENTATION	0.000000	0.381443	0.007752	0.000000
REL DIFF ORDER	0.000000	0.404255	0.005348	0.000000
REL GAP ERROR 100	0.043165	0.904255	0.155080	0.241758
REL GAP ERROR 500	0.035971	0.819149	0.096257	0.131868
REL GAP ERROR 1000	0.031175	0.776596	0.088235	0.098901
REL GAP ERROR 10000	0.004796	0.648936	0.048128	0.087912

Table B.2: Complete table of run shown in table 6.1 on page 44

NCUT	1000	1000	1000	1000
NGUIDES	1	1	1	1
WINDOWSIZE	3	3	3	3
THRESHOLD	1	1	1	1
TARGET GENOME	Escherichia coli	Pseudoxanthomonas suwonensis	Rhodobacter sphaeroides	Staphylococcus aureus
GENOME SIZE	4639675	3419049	4603060	2903081
NCONTIGS	481	303	583	162
MIN CONTIG LENGTH	152	151	205	203
MAX CONTIG LENGTH	73062	90572	60714	169214
NSCAFFOLDS	19	27	64	17
MIN SCAFFOLD LENGTH	1800	7786	1205	22913
MAX SCAFFOLD LENGTH	1196719	182780	392693	556033
COVERAGE	1.02	1.19	1.08	1.21
N50 CONTIGS	19872	26043	15801	52792
N50 SCAFFOLDS	597757	46936	84362	252200
N CONTIGS USED	433	65	389	98
REL CONTIGS USED	.900	.214	.667	.604
N PAIRS	414	38	325	81
DIFF CHROMOSOMES	0	0	2	0
DIFF ORIENTATION	0	6	0	3
DIFF ORDER	0	5	1	2
GAP ERROR 100	16	34	53	31
GAP ERROR 500	13	27	43	24
GAP ERROR 1000	12	26	40	21
GAP ERROR 10000	6	21	17	16
REL DIFF CHROMOSOMES	0.000000	0.000000	0.005141	0.000000
REL DIFF ORIENTATION	0.000000	0.092308	0.000000	0.030612
REL DIFF ORDER	0.000000	0.131579	0.003077	0.024691
REL GAP ERROR 100	0.038647	0.894737	0.163077	0.382716
REL GAP ERROR 500	0.031401	0.710526	0.132308	0.296296
REL GAP ERROR 1000	0.028986	0.684211	0.123077	0.259259
REL GAP ERROR 10000	0.014493	0.552632	0.052308	0.197531

Table B.3: Complete table of run shown in table 6.2 on page 45

NCUT	0	0	0	0	0	0	0	0	0	0	0	0	0
NGUIDES	10	10	10	10	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2	2	2	2	2
THRESHOLD	1	2	3	4	5	6	7	8	9	10			
GENOME SIZE	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675
NCONTIGS	481	481	481	481	481	481	481	481	481	481	481	481	481
MIN CONTIG LENGTH	152	152	152	152	152	152	152	152	152	152	152	152	152
MAX CONTIG LENGTH	73062	73062	73062	73062	73062	73062	73062	73062	73062	73062	73062	73062	73062
NSCAFFOLDS	16	8	21	32	53	70	75	71	68	56			
MIN SCAFFOLD LENGTH	525	652	877	877	1384	1384	2623	2623	355	355			
MAX SCAFFOLD LENGTH	3707904	2627065	1146682	923059	380397	232659	163998	134866	123232	78547			
COVERAGE	1	1.03	1	1.02	1.03	1.02	0.99	0.98	0.97	0.97			
N50 CONTIGS	19872	19872	19872	19872	19872	19872	19872	19872	19872	19872			
N50 SCAFFOLDS	3707904	2627065	587190	332993	102239	50979	35364	34312	32967	24036			
N CONTIGS USED	456	438	419	397	327	289	248	227	201	139			
REL CONTIGS USED	0.948	0.91	0.871	0.825	0.679	0.6	0.515	0.471	0.417	0.288			
N PAIRS	440	430	398	365	274	219	173	156	133	83			
DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0	0			
DIFF ORIENTATION	49	0	0	0	0	0	0	0	0	0			
DIFF ORDER	46	2	0	0	0	0	0	0	0	0			
GAP ERROR 100	47	34	29	32	31	23	11	8	10	3			
GAP ERROR 500	32	22	18	21	24	17	8	6	4	0			
GAP ERROR 1000	24	14	11	14	19	12	5	4	3	0			
GAP ERROR 10000	9	5	2	2	2	0	0	0	0	0			
REL DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0	0			
REL DIFF ORIENTATION	0.107456	0	0	0	0	0	0	0	0	0			
REL DIFF ORDER	0.104545	0.004651	0	0	0	0	0	0	0	0			
REL GAP ERROR 100	0.106818	0.07907	0.072864	0.087671	0.113139	0.105023	0.063584	0.051282	0.075188	0.036145			
REL GAP ERROR 500	0.072727	0.051163	0.045226	0.057534	0.087591	0.077626	0.046243	0.038462	0.030075	0			
REL GAP ERROR 1000	0.054545	0.032558	0.027638	0.038356	0.069343	0.054795	0.028902	0.025641	0.022556	0			
REL GAP ERROR 10000	0.020455	0.011628	0.005025	0.005479	0.007299	0	0	0	0	0			

Table B.4: Increasing threshold for E. coli without contig end extraction.

NCUT	0	0	0	0	0	0	0	0	0	0
NGUIDES	10	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2	2
THRESHOLD	1	2	3	4	5	6	7	8	9	10
GENOME SIZE	3419049	3419049	3419049	3419049	3419049	3419049	3419049	3419049	3419049	3419049
NCONTIGS	303	303	303	303	303	303	303	303	303	303
MIN CONTIG LENGTH	151	151	151	151	151	151	151	151	151	151
MAX CONTIG LENGTH	90572	90572	90572	90572	90572	90572	90572	90572	90572	90572
NSCAFFOLDS	22	13	16	24	32	29	26	15	6	2
MIN SCAFFOLD LENGTH	649	1553	17250	5164	5164	5164	3845	10372	4165	7120
MAX SCAFFOLD LENGTH	1078268	1443141	608178	430518	353217	155400	155400	155400	112757	16158
COVERAGE	1.3	1.61	1.52	1.4	1.3	1.13	1.08	1.04	0.99	0.99
N50 CONTIGS	26043	26043	26043	26043	26043	26043	26043	26043	26043	26043
N50 SCAFFOLDS	195757	521215	353978	153907	75643	47935	39003	31424	26987	26043
N CONTIGS USED	162	141	133	119	105	81	66	39	13	5
REL CONTIGS USED	0.534	0.465	0.438	0.392	0.346	0.267	0.217	0.128	0.042	0.016
N PAIRS	140	128	117	95	73	52	40	24	7	3
DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0	0
DIFF ORIENTATION	45	41	39	30	17	12	8	2	1	0
DIFF ORDER	49	44	37	26	10	7	3	0	0	0
GAP ERROR 100	122	118	108	86	65	46	35	17	3	0
GAP ERROR 500	114	111	101	82	62	43	32	15	2	0
GAP ERROR 1000	107	102	94	75	56	39	28	11	2	0
GAP ERROR 10000	83	79	72	60	40	28	19	6	1	0
REL DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0	0
REL DIFF ORIENTATION	0.277778	0.29078	0.293233	0.252101	0.161905	0.148148	0.121212	0.051282	0.076923	0
REL DIFF ORDER	0.35	0.34375	0.316239	0.273684	0.136986	0.134615	0.075	0	0	0
REL GAP ERROR 100	0.871429	0.921875	0.923077	0.905263	0.890411	0.884615	0.875	0.708333	0.428571	0
REL GAP ERROR 500	0.814286	0.867188	0.863248	0.863158	0.849315	0.826923	0.8	0.625	0.285714	0
REL GAP ERROR 1000	0.764286	0.796875	0.803419	0.789474	0.767123	0.75	0.7	0.458333	0.285714	0
REL GAP ERROR 10000	0.592857	0.617188	0.615385	0.631579	0.547945	0.538462	0.475	0.25	0.142857	0

Table B.5: Increasing threshold for P. suwonensis without contig end extraction.

NCUT	0	0	0	0	0	0	0	0	0	0	0	0	0
NGUIDES	10	10	10	10	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2	2	2	2	2
THRESHOLD	1	2	3	4	5	6	7	8	9	10			
GENOME SIZE	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060
NCONTIGS	583	583	583	583	583	583	583	583	583	583	583	583	583
MIN CONTIG LENGTH	205	205	205	205	205	205	205	205	205	205	205	205	205
MAX CONTIG LENGTH	60714	60714	60714	60714	60714	60714	60714	60714	60714	60714	60714	60714	60714
NSCAFFOLDS	43	50	74	28	10	3	1	0	0	0	0	0	0
MIN SCAFFOLD LENGTH	507	690	690	3748	6788	11968	11968						
MAX SCAFFOLD LENGTH	1314966	299150	80375	59403	55987	18022	11968						
COVERAGE	1.35	1.03	0.97	0.98	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
N50 CONTIGS	15801	15801	15801	15801	15801	15801	15801	15801	15801	15801	15801	15801	15801
N50 SCAFFOLDS	404156	84439	21151	17378	16746	16072	15801	15801	15801	15801	15801	15801	15801
N CONTIGS USED	414	325	197	58	21	6	2	0	0	0	0	0	0
REL CONTIGS USED	0.71	0.557	0.337	0.099	0.036	0.01	0.003	0	0	0	0	0	0
N PAIRS	371	275	123	30	11	3	1	0	0	0	0	0	0
DIFF CHROMOSOMES	22	2	1	0	0	0	0	0	0	0	0	0	0
DIFF ORIENTATION	111	0	0	0	0	0	0	0	0	0	0	0	0
DIFF ORDER	99	1	0	0	0	0	0	0	0	0	0	0	0
GAP ERROR 100	139	42	19	11	4	1	0	0	0	0	0	0	0
GAP ERROR 500	110	22	8	5	3	1	0	0	0	0	0	0	0
GAP ERROR 1000	97	15	3	1	1	0	0	0	0	0	0	0	0
GAP ERROR 10000	76	5	1	0	0	0	0	0	0	0	0	0	0
REL DIFF CHROMOSOMES	0.05314	0.006154	0.005076	0	0	0	0	0	0	0	0	0	0
REL DIFF ORIENTATION	0.268116	0	0	0	0	0	0	0	0	0	0	0	0
REL DIFF ORDER	0.266846	0.003636	0	0	0	0	0	0	0	0	0	0	0
REL GAP ERROR 100	0.374663	0.152727	0.154472	0.366667	0.363636	0.333333	0	0	0	0	0	0	0
REL GAP ERROR 500	0.296496	0.08	0.065041	0.166667	0.272727	0.333333	0	0	0	0	0	0	0
REL GAP ERROR 1000	0.261456	0.054545	0.02439	0.033333	0.090909	0	0	0	0	0	0	0	0
REL GAP ERROR 10000	0.204852	0.018182	0.00813	0	0	0	0	0	0	0	0	0	0

Table B.6: Increasing threshold for R. sphaeroides without contig end extraction.

NCUT	0	0	0	0	0	0	0	0	0	0
NGUIDES	10	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2	2
THRESHOLD	1	2	3	4	5	6	7	8	9	10
GENOME SIZE	2903081	2903081	2903081	2903081	2903081	2903081	2903081	2903081	2903081	2903081
NCONTIGS	162	162	162	162	162	162	162	162	162	162
MIN CONTIG LENGTH	203	203	203	203	203	203	203	203	203	203
MAX CONTIG LENGTH	169214	169214	169214	169214	169214	169214	169214	169214	169214	169214
NSCAFFOLDS	11	12	14	12	15	16	15	12	10	6
MIN SCAFFOLD LENGTH	959	959	1896	1896	1896	2083	939	939	939	12629
MAX SCAFFOLD LENGTH	1982144	1232112	1085175	840011	840011	360044	294590	137858	122999	106287
COVERAGE	1.11	1.16	1.07	1.07	1.08	1.06	0.98	0.98	0.98	.98
N50 CONTIGS	52792	52792	52792	52792	52792	52792	52792	52792	52792	52792
N50 SCAFFOLDS	1982144	511755	511755	336786	232045	107884	84881	72939	57025	54353
N CONTIGS USED	131	106	100	93	87	59	48	31	24	14
REL CONTIGS USED	0.808	0.654	0.617	0.574	0.537	0.364	0.296	0.191	0.148	.086
N PAIRS	120	94	86	81	72	43	33	19	14	8
DIFF CHROMOSOMES	3	0	0	0	0	0	0	0	0	0
DIFF ORIENTATION	1	1	0	0	0	0	0	0	0	0
DIFF ORDER	0	0	0	0	0	0	0	0	0	0
GAP ERROR 100	27	26	15	14	12	7	8	5	3	2
GAP ERROR 500	13	16	7	6	4	2	2	1	1	1
GAP ERROR 1000	10	11	2	2	2	1	1	0	0	0
GAP ERROR 10000	6	7	1	1	1	0	1	0	0	0
REL DIFF CHROMOSOMES	0.022901	0	0	0	0	0	0	0	0	0.000000
REL DIFF ORIENTATION	0.007634	0.009434	0	0	0	0	0	0	0	0.000000
REL DIFF ORDER	0	0	0	0	0	0	0	0	0	0.000000
REL GAP ERROR 100	0.225	0.276596	0.174419	0.17284	0.166667	0.162791	0.242424	0.263158	0.214286	0.250000
REL GAP ERROR 500	0.108333	0.170213	0.081395	0.074074	0.055556	0.046512	0.060606	0.052632	0.071429	0.125000
REL GAP ERROR 1000	0.083333	0.117021	0.023256	0.024691	0.027778	0.023256	0.030303	0	0	0.000000
REL GAP ERROR 10000	0.05	0.074468	0.011628	0.012346	0.013889	0	0.030303	0	0	0.000000

Table B.7: Increasing threshold for S. aureus without contig end extraction.

NCUT	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
NGUIDES	10	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2	2
THRESHOLD	1	2	3	4	5	6	7	8	9	10
GENOME SIZE	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675	4639675
NCONTIGS	481	481	481	481	481	481	481	481	481	481
MIN CONTIG LENGTH	152	152	152	152	152	152	152	152	152	152
MAX CONTIG LENGTH	73062	73062	73062	73062	73062	73062	73062	73062	73062	73062
NSCAFFOLDS	21	25	41	49	57	55	52	49	42	28
MIN SCAFFOLD LENGTH	388	388	388	388	388	388	2537	2537	2537	977
MAX SCAFFOLD LENGTH	2617579	1413486	483236	405869	379541	256224	149161	149161	149161	75741
COVERAGE	1	1	1.02	1	0.97	0.97	0.98	0.97	0.98	0.98
N50 CONTIGS	19872	19872	19872	19872	19872	19872	19872	19872	19872	19872
N50 SCAFFOLDS	2617579	468635	171828	104996	58409	37464	31532	31326	31326	23334
N CONTIGS USED	424	426	378	333	279	223	171	154	141	71
REL CONTIGS USED	0.881	0.885	0.785	0.692	0.58	0.463	0.355	0.32	0.293	0.147
N PAIRS	403	401	337	284	222	168	119	105	99	43
DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0	0
DIFF ORIENTATION	59	0	0	0	0	0	0	0	0	0
DIFF ORDER	57	2	0	0	0	0	0	0	0	0
GAP ERROR 100	40	29	21	20	16	8	7	8	9	2
GAP ERROR 500	25	17	14	15	10	6	6	6	7	1
GAP ERROR 1000	20	12	10	11	7	4	4	4	4	1
GAP ERROR 10000	7	2	3	3	0	0	0	0	0	0
REL DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0	0
REL DIFF ORIENTATION	0.139151	0	0	0	0	0	0	0	0	0
REL DIFF ORDER	0.141439	0.004988	0	0	0	0	0	0	0	0
REL GAP ERROR 100	0.099256	0.072319	0.062315	0.070423	0.072072	0.047619	0.058824	0.07619	0.090909	0.046512
REL GAP ERROR 500	0.062035	0.042394	0.041543	0.052817	0.045045	0.035714	0.05042	0.057143	0.070707	0.023256
REL GAP ERROR 1000	0.049628	0.029925	0.029674	0.038732	0.031532	0.02381	0.033613	0.038095	0.040404	0.023256
REL GAP ERROR 10000	0.01737	0.004988	0.008902	0.010563	0	0	0	0	0	0

Table B.8: Increasing threshold for E. coli with contig end length $N = 1,000$.

NCUT	1000	1000	1000	1000	1000	1000	1000	1000	1000
NGUIDES	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2
THRESHOLD	2	3	4	5	6	7	8	9	10
GENOME SIZE	3419049	3419049	3419049	3419049	3419049	3419049	3419049	3419049	3419049
NCONTIGS	303	303	303	303	303	303	303	303	303
MIN CONTIG LENGTH	151	151	151	151	151	151	151	151	151
MAX CONTIG LENGTH	90572	90572	90572	90572	90572	90572	90572	90572	90572
NSCAFFOLDS	34	28	28	22	19	17	12	6	3
MIN SCAFFOLD LENGTH	1362	1233	1233	10922	9079	6755	3312	6755	19737
MAX SCAFFOLD LENGTH	262476	398749	198793	263414	137267	137267	115326	84419	60460
COVERAGE	1.15	1.2	1.11	1.13	1.04	1.04	1.01	0.99	0.99
N50 CONTIGS	26043	26043	26043	26043	26043	26043	26043	26043	26043
N50 SCAFFOLDS	45417	46037	39003	35902	32397	30833	27373	26043	27041
N CONTIGS USED	95	74	65	49	47	39	25	12	6
REL CONTIGS USED	0.313	0.244	0.214	0.161	0.155	0.128	0.082	0.039	0.019
N PAIRS	61	46	37	27	28	22	13	6	3
DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0
DIFF ORIENTATION	12	10	9	4	3	1	0	0	0
DIFF ORDER	5	6	2	1	1	0	0	0	0
GAP ERROR 100	38	34	26	20	15	13	7	2	1
GAP ERROR 500	31	29	21	16	12	9	6	2	1
GAP ERROR 1000	28	27	20	15	9	7	3	1	1
GAP ERROR 10000	21	18	15	10	6	4	1	0	0
REL DIFF CHROMOSOMES	0	0	0	0	0	0	0	0	0
REL DIFF ORIENTATION	0.126316	0.135135	0.138462	0.081633	0.06383	0.025641	0	0	0
REL DIFF ORDER	0.081967	0.130435	0.054054	0.037037	0.035714	0	0	0	0
REL GAP ERROR 100	0.622951	0.73913	0.702703	0.740741	0.535714	0.590909	0.538462	0.333333	0.333333
REL GAP ERROR 500	0.508197	0.630435	0.567568	0.592593	0.428571	0.409091	0.461538	0.333333	0.333333
REL GAP ERROR 1000	0.459016	0.586957	0.540541	0.555556	0.321429	0.318182	0.230769	0.166667	0.333333
REL GAP ERROR 10000	0.344262	0.391304	0.405405	0.37037	0.214286	0.181818	0.076923	0	0

Table B.9: Increasing threshold for P. suwonensis with contig end length $N = 1{,}000$.

NCUT	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
NGUIDES	10	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2	2
THRESHOLD	1	2	3	4	5	6	7	8	9	10
GENOME SIZE	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060	4603060
NCONTIGS	583	583	583	583	583	583	583	583	583	583
MIN CONTIG LENGTH	205	205	205	205	205	205	205	205	205	205
MAX CONTIG LENGTH	60714	60714	60714	60714	60714	60714	60714	60714	60714	60714
NSCAFFOLDS	55	53	44	17	10	5	4	1	1	0
MIN SCAFFOLD LENGTH	507	1336	690	690	4200	7384	9227	9046	9046	
MAX SCAFFOLD LENGTH	593208	273799	110861	52226	42507	24564	24458	9046	9046	
COVERAGE	1.07	1.02	0.98	0.98	0.97	0.97	0.97	0.97	0.97	0.97
N50 CONTIGS	15801	15801	15801	15801	15801	15801	15801	15801	15801	15801
N50 SCAFFOLDS	158094	49298	21151	17005	16684	15801	15801	15801	15801	15801
N CONTIGS USED	404	270	129	41	22	10	8	2	2	0
REL CONTIGS USED	0.692	0.463	0.221	0.07	0.037	0.017	0.013	0.003	0.003	0
N PAIRS	349	217	85	24	12	5	4	1	1	0
DIFF CHROMOSOMES	13	0	1	1	0	0	0	0	0	0
DIFF ORIENTATION	63	5	0	0	0	0	0	0	0	0
DIFF ORDER	53	6	0	0	0	0	0	0	0	0
GAP ERROR 100	82	27	8	3	1	1	0	0	0	0
GAP ERROR 500	60	16	4	2	0	0	0	0	0	0
GAP ERROR 1000	54	13	4	2	0	0	0	0	0	0
GAP ERROR 10000	42	6	2	1	0	0	0	0	0	0
REL DIFF CHROMOSOMES	0.032178	0	0.007752	0.02439	0	0	0	0	0	0
REL DIFF ORIENTATION	0.155941	0.018519	0	0	0	0	0	0	0	0
REL DIFF ORDER	0.151862	0.02765	0	0	0	0	0	0	0	0
REL GAP ERROR 100	0.234957	0.124424	0.094118	0.125	0.083333	0.2	0	0	0	0
REL GAP ERROR 500	0.17192	0.073733	0.047059	0.083333	0	0	0	0	0	0
REL GAP ERROR 1000	0.154728	0.059908	0.047059	0.083333	0	0	0	0	0	0
REL GAP ERROR 10000	0.120344	0.02765	0.023529	0.041667	0	0	0	0	0	0

Table B.10: Increasing threshold for R. sphaeroides with contig end length $N = 1,000$.

NCUT	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
NGUIDES	10	10	10	10	10	10	10	10	10	10
WINDOWSIZE	2	2	2	2	2	2	2	2	2	2
THRESHOLD	1	2	3	4	5	6	7	8	9	10
GENOME SIZE	2903081	2903081	2903081	2903081	2903081	2903081	2903081	2903081	2903081	2903081
NCONTIGS	162	162	162	162	162	162	162	162	162	162
MIN CONTIG LENGTH	203	203	203	203	203	203	203	203	203	203
MAX CONTIG LENGTH	169214	169214	169214	169214	169214	169214	169214	169214	169214	169214
NSCAFFOLDS	17	14	20	18	17	16	16	12	11	9
MIN SCAFFOLD LENGTH	2217	5342	463	1210	1210	4172	13038	5072	939	2546
MAX SCAFFOLD LENGTH	809679	886347	255257	255257	255257	255257	255257	255257	249406	249406
COVERAGE	1.07	1.12	1	1.01	1.01	0.99	0.99	0.97	0.96	.96
N50 CONTIGS	52792	52792	52792	52792	52792	52792	52792	52792	52792	52792
N50 SCAFFOLDS	284993	171854	105031	113389	105031	74365	69629	75248	68207	66100
N CONTIGS USED	118	91	74	73	69	57	50	37	27	22
REL CONTIGS USED	0.728	0.561	0.456	0.45	0.425	0.351	0.308	0.228	0.166	.135
N PAIRS	101	77	54	55	52	41	34	25	16	13
DIFF CHROMOSOMES	3	1	0	0	0	0	0	0	0	0
DIFF ORIENTATION	24	10	0	0	0	0	0	0	0	0
DIFF ORDER	21	7	0	0	0	0	0	0	0	0
GAP ERROR 100	30	28	12	12	12	5	3	2	0	0
GAP ERROR 500	20	18	5	6	6	2	2	1	0	0
GAP ERROR 1000	19	16	3	5	5	2	2	1	0	0
GAP ERROR 10000	10	9	3	4	4	2	2	1	0	0
REL DIFF CHROMOSOMES	0.025424	0.010989	0	0	0	0	0	0	0	0.000000
REL DIFF ORIENTATION	0.20339	0.10989	0	0	0	0	0	0	0	0.000000
REL DIFF ORDER	0.207921	0.090909	0	0	0	0	0	0	0	0.000000
REL GAP ERROR 100	0.29703	0.363636	0.222222	0.218182	0.230769	0.121951	0.088235	0.08	0	0.000000
REL GAP ERROR 500	0.19802	0.233766	0.092593	0.109091	0.115385	0.04878	0.058824	0.04	0	0.000000
REL GAP ERROR 1000	0.188119	0.207792	0.055556	0.090909	0.096154	0.04878	0.058824	0.04	0	0.000000
REL GAP ERROR 10000	0.09901	0.116883	0.055556	0.072727	0.076923	0.04878	0.058824	0.04	0	0.000000

Table B.11: Increasing threshold for S. aureus with contig end length $N = 1,000$.